

The University of Akron

IdeaExchange@UAkron

---

Williams Honors College, Honors Research  
Projects

The Dr. Gary B. and Pamela S. Williams Honors  
College

---

Spring 2023

## Wireless Environmental Weather Monitor

Joel Christie-Millett  
jsm157@uakron.edu

Nathan Schroeder  
nes39@uakron.edu

Sylvester Wilson  
sw120@uakron.edu

Matthew Szijarto  
ms473@uakron.edu

Follow this and additional works at: [https://ideaexchange.uakron.edu/honors\\_research\\_projects](https://ideaexchange.uakron.edu/honors_research_projects)



Part of the [Electrical and Electronics Commons](#), [Power and Energy Commons](#), [Systems and Communications Commons](#), and the [VLSI and Circuits, Embedded and Hardware Systems Commons](#)

Please take a moment to share how this work helps you [through this survey](#). Your feedback will be important as we plan further development of our repository.

---

### Recommended Citation

Christie-Millett, Joel; Schroeder, Nathan; Wilson, Sylvester; and Szijarto, Matthew, "Wireless Environmental Weather Monitor" (2023). *Williams Honors College, Honors Research Projects*. 1641. [https://ideaexchange.uakron.edu/honors\\_research\\_projects/1641](https://ideaexchange.uakron.edu/honors_research_projects/1641)

This Dissertation/Thesis is brought to you for free and open access by The Dr. Gary B. and Pamela S. Williams Honors College at IdeaExchange@UAkron, the institutional repository of The University of Akron in Akron, Ohio, USA. It has been accepted for inclusion in Williams Honors College, Honors Research Projects by an authorized administrator of IdeaExchange@UAkron. For more information, please contact [mjon@uakron.edu](mailto:mjon@uakron.edu), [uapress@uakron.edu](mailto:uapress@uakron.edu).

**Honors Research Project**  
**Satellite Weather Station (SDP-DT11)**  
**Individual Student Contributions**

Joel Christie-Millett (Honors)

My contribution to this project was to design and implement the electrical power distribution system. The goal was to make a system that could monitor weather conditions for extended periods of time without any human intervention. I designed an efficient power system to achieve this. The device operates from battery power and the battery charges from renewable solar energy captured by a solar panel. The rest of the device was implemented in such a way that it consumed as little power as possible while still providing accurate data. I designed a system that only powers each individual subsystem when it is needed so that it doesn't consume any excess power. In the design process, I guided Matt on the power requirements of the sensors. I also worked with Sylvester and Nate on some options for power-efficient sleep mode options for the microcontroller and communication systems. On a more specific scale, I implemented and troubleshot battery charging integrated circuits (IC's), voltage regulator IC's, and transistors. I also dealt with the physical wiring of the prototype before we pursued a printed circuit board. Finally, I completed power harvesting and consumption calculations to determine how long the system can last without human interaction.

Sylvester Wilson, Jr.

My contribution to this project was to design and implement a program that would take values recorded by the weather monitor, use said values in an algorithm to calculate the given Fine Dead Fuel Moisture (FDFM) and Probability of Ignition (PIG) for the given environment, and to display all this information to an application for an end-user to view. Data, in the form of short messages, were sent from the weather station to Swarm© servers via GPS. The program would pull said messages from the servers down to my personal laptop in the form of text files. These text files were then parsed and had their values stored in the associated variables for temperature, humidity, barometric pressure, GPS coordinates, etc. These values were then used to calculate FDFM and PIG. Once this process is complete, the information is then sent to an application called Ubidots, which is a website that specializes in condition monitoring and has an intuitive user interface. The data displayed is updated in accordance with the execution of the program, which can run indefinitely.

### Nate Schroeder

As a computer engineering student and Project Manager of the senior design team, my contribution to the project was twofold. First, my sub system as a computer engineering student was the communication system and embedded firmware development. The communication side of the project comprised of the primary satellite communication system the team would be using along with the lower level intra board communication schemes that allowed all the sensors to talk with the host microcontroller. I wrote much of the embedded firmware that combined all the other sub systems together to allow for smooth communication and data transmission. I then combined all the subsystems into a schematic and printed circuit board (PCB) using an ECAD tool called KiCAD which allowed us to take our breadboarded design and get a professional PCB manufactured. The second task I was elected for was to be the Project Manager of the project. This meant that I oversaw the budget, maintained the project timeline, ordered parts, and scheduled meetings with our Faculty Advisors. The overall cooperation and dedication of each team member ultimately lead to a successful senior design project.

### Matthew Szijarto (Graduating with Honors Requirements)

My contribution to this project was to design and implement a set of sensors that would meet the goal of early detection and monitoring of a fire. To do this, research was done in the first semester to determine what attributes of a fire needed to be monitored to determine its severity and direction of spread. As a part of our engineering requirements, it was stated that two metrics would be used to determine if the system was in a fire-prone situation; fine dead fuel moisture and probability of ignition. These need three total sensors: light, temperature, and humidity. As for other sensors, the monitoring of the wind speed and air pressure could be used to determine even further if conditions were poor, on a human reading basis. After discussing more sensors, but pairing down for budget reasons, five sensors were decided: wind speed, temperature, humidity, air pressure, and light. These sensors were chosen based on power requirements given by Joel, for the batteries and solar panel, and also to interface properly with the CPU chosen: choosing multiple I2C and one analog voltage sensor made integration easiest. Finally, I helped code the sensors to integrate properly with the circuit. I also helped with integration and testing of the power circuit design.

# Weather Monitoring Station

## Project Design Report

DT11

Joel Christie-Millett

Matthew Szijarto

Sylvester Wilson

Nathan Schroeder

Huu Nghi Tran

11/28/2022

## Table of Contents

List of Figures .....	6
List of Tables .....	7
Abstract .....	8
1. Problem Statement .....	9
1.1. Need .....	9
1.2. Objective .....	9
1.3. Background .....	10
1.4. Marketing Requirements .....	17
2. Engineering Analysis .....	18
2.1. Circuits .....	18
2.1.1 Power Distribution.....	18
2.2. Electronics.....	29
2.2.1 Microcontroller .....	29
2.2.2 Sensors.....	31
2.3. Mechanical .....	34
2.4. Communications.....	36
2.4.1 Wired Communications .....	36
2.4.2 Wireless Communications .....	40
2.6. Computer Networks and Software .....	44
2.7. Embedded Systems .....	48
2.7.1 External Storage Module .....	48
2.7.2 Global Positioning Module.....	49
3. Engineering Requirements Specification.....	50
4. Engineering Standards Specification .....	52
5. Accepted Technical Design .....	54
5.1. Hardware Design.....	54
5.2 Software Design .....	71
6. Mechanical Design.....	76
7. Team Information .....	78

8. Parts List .....	78
8.1. Parts List for Accepted Technical Design.....	78
8.2. Materials Budget .....	79
9. Project Schedules .....	81
10. Conclusion .....	84
11. References .....	85
12. Appendices.....	87

## List of Figures

Figure 1: Probability of Ignition Table .....	12
Figure 2: Fine Dead Fuel Moisture Table .....	13
Figure 3: Example Power Calculations.....	20
Figure 4: Example Power Calculations Cont.....	21
Figure 5: Peak Sun Hours Map .....	24
Figure 6: Preliminary Power Circuit Diagram.....	26
Figure 8: Example I2C Communication to SHT31 Temperature Sensor .....	37
Figure 9: Displaying SHT31 Temperature and Humidity Data.....	38
Figure 10: Displaying LPS33HW Pressure Sensor Data.....	39
Figure 11: Example Communication Flow .....	40
Figure 12: Swarm “Cube-Sat” Satellites.....	42
Figure 13: Antenna Implementation for Testing .....	43
Figure 14: User Interface Example .....	45
Figure 15: Example of FDFM Calculation Tests.....	47
Figure 16: Level 0 Hardware Block Diagram.....	54
Figure 17: Level 1 Hardware Block Diagram.....	56
Figure 18: Level 2 Sensor Block Diagram .....	58
Figure 19: PIC24FJ128GA010 Microcontroller Schematic .....	59
Figure 20: Sensors Control and Connection Circuit .....	60
Figure 21: Level 2 Communication Block Diagram.....	61
Figure 22: Swarm Modem Circuit Layout.....	62
Figure 23: Level 2 Power Block Diagram .....	64
Figure 24: Level 3 Power Block Diagram .....	66
Figure 25: Level 4 Power Block Diagram .....	68
Figure 26: Power and Voltage Regulation Sub-System Schematic.....	69
Figure 27: Software Level 0 Block Diagram.....	71
Figure 28: Software Level 1 Block Diagram.....	73
Figure 29: Software High Level Flowchart .....	75
Figure 30: Mechanical Drawing with Estimated Dimensions. ....	76
Figure 31: Mechanical 3D Rendering .....	77
Figure 32: Semester 1 Project Schedule .....	81

Figure 33: Project Schedule cont. ....	82
Figure 34: Project Schedule cont. ....	83
Figure 35: Swarm M138 Satellite Modem.....	87
Figure 36: Alternative 3D View .....	88
Figure 37: Interior Dimensions.....	89

## List of Tables

Table 1: Example Power Consumption .....	22
Table 2: Estimated Weight of Major Components .....	35
Table 3: Fine Dead Fuel Moisture and Probability of Ignition Calculator Pseudocode .....	46
Table 4: Engineering Requirements.....	50
Table 5: Engineering Standards Implemented.....	52
Table 6: Functional Requirement Table for the Level 0 Hardware Block Diagram .....	54
Table 7: Functional Requirement Table for the Level 1 Hardware Block Diagram .....	56
Table 8: Functional Requirement Table for the Level 2 Sensor Block Diagram .....	58
Table 9: Functional Requirement Table for the Level 2 Communication Block Diagram .....	61
Table 10: Functional Requirement Table for the Level 2 Power Block Diagram .....	64
Table 11: Functional Requirement Table for the Level 3 Power Block Diagram .....	66
Table 12: Functional Requirement Table for the Level 4 Power Block Diagram .....	68
Table 13: Functional Requirement Table for the Level 0 Software Block Diagram.....	72
Table 14: Functional Requirement Table for the Level 1 Software Block Diagram.....	74
Table 15: Parts List .....	78
Table 16: Materials Budget.....	79
Table 17: PIC24FJ129GA010 I2C Code Library .....	90
Table 18: NMEA Checksum Code .....	91
Table 19: Raw Pressure to hPa Code.....	91
Table 20: Raw Temperature to °F.....	92
Table 21: Raw Humidity to Relative Percentage.....	92
Table 22: Demonstration Code .....	92



## Abstract

[JCM, MS, SW, NS]

Forest fires are an ever-present problem in some regions of the world and the methods of detecting them are antiquated. This report aims to outline the process of creating a device that can detect forest fire conditions and notify a user. This device senses several environmental conditions, such as temperature, relative humidity, barometric pressure, wind speed, and sunlight, and processes them using an onboard microcontroller to determine the likelihood of a fire. These parameters were chosen due to the common use and reliability of the Probability of Ignition chart (PIG). This device is designed to be deployed in environments that do not have existing infrastructure and will not be interacted with for an entire fire season. To achieve this the device uses solar energy harvesting, due to its relative abundance in the regions where fires are the largest problem. The device communicates with the user using satellite communication due to its global versatility. A few important factors in this design were energy efficiency, sensor accuracy, a robust and accurate implementation of a fire detection algorithm, and finally fast communication to the end user.

### Key Features:

- Satellite Communication
- Solar Energy Harvesting
- Accurate detection of fire conditions
- Deployable in rugged terrain for an entire fire season

# **1. Problem Statement**

## **1.1. Need**

[NS]

The ever-adapting global climate yields drastic changes in weather patterns throughout the United States and rest of the world [11]. These changes result in more severe tornadoes, tropical storms, and even droughts [11]. Furthermore, more frequent droughts are leading to an increase in wildfires specifically in the western part of the United States. According to the Environmental Protection Agency (EPA); of the 10 years with the largest acreage burned, all have occurred since 2004, including the peak year in 2015 [11]. Therefore, as wildfires become more prevalent in the world, a real time early warning and detection system is needed. This system could provide firefighters and homeowners with real time data of what conditions are like in the forest during wildfire season and more time to evacuate if needed. This increased situational awareness can be instrumental in saving lives.

## **1.2. Objective**

[NS]

The objective of this project is to monitor the conditions of the environment for the goal of early wildfire prevention and situational awareness. The device will be able to communicate with users, using a remote application or website, to display the status of the environment and wildfire probability at the sensor's location. In turn, the application will notify users if the conditions constitute the possibility of an emergency or are favorable for wildfire conditions. Multiple of these devices can be wirelessly connected to provide users greater accuracy, range, and awareness of remote environments, as well as providing a safety net in case one sensor is

destroyed. This will allow for redundancy in the case of life-threatening emergencies and allow the users the most accurate and up to date information regarding the environment around them.

### **1.3. Background**

[NS]

The basic theory behind the concept being proposed, the Wireless Environmental Weather Monitoring Station, is a remote system to provide measurements of the environment around the sensor and then use those measurements to determine if conditions could result in a wildfire. The basic concept behind the proposal is the utilization of a wireless network, capable of providing interconnectivity between devices where no network infrastructure is already pre-established. Wireless mesh networks are defined by Ian F. Akyildiz and others as being “dynamically self-organized and self-configured, with the nodes in the network automatically establishing and maintaining mesh connectivity among themselves” [1]. Each node is composed of mesh routers and mesh clients, resulting in every node having the option to function as both a host and router. For clarification, a router is a node that has access to two or more networks, whereas a host is a node that allows for the accessing of information of all the other nodes via a user interface/software or by some other means [2]. This allows for packets of data to be sent from nodes on behalf of other nodes that may not be within direct transmission range of their destination [1]. That form of transmitting data is referred to as multi-hopping and allows for a wireless mesh router to maintain a similar range to a “conventional” router when it would be smaller otherwise.

There are further advantages to using a wireless mesh network, one of which being increased mobility of nodes that primarily act as mesh clients. Having the capability to freely adjust the location of the sensors while maintaining a connection to the network is pivotal to the

functionality of the proposed concept. Another advantage to using the network is its flexibility to be integrated to other pre-existing networks [1]. Connectivity to the Internet, the Global Navigation Satellite System, cellular devices, and Bluetooth are all aspects that are being considered in the proposal and are made possible through the mesh network.

[MS, JCM]

According to a 2020 research paper regarding the AI detection of smoke and fire, the usual detection method of a fire is outlined: “the first notification of a fire is typically received from someone calling 9-1-1 (the United States nation-wide emergency number) from their mobile phone. When asked, fire personnel tend to say that fires are typically reported within 15 min of ignition.” [3] After this 15-minute response time, firefighters typically communicate with a home base regarding proper response to the fire. According to a graphic by the website for the California Department of Forestry and Fire Protection, a high-tech application used for communication between firefighters is a “Mobile Communication Center” - a truck with computer stations, local weather reports, and HAM/AM/FM radio transmitters/receivers [4].

[NS]

Current detection and communication schemes are mostly person-to-person; there are limitations to the current technology behind fire protection. First and most obvious is the delay of first detection. Initial delay between the start of the fire and first civilian response is about 15 minutes. Moreover, the paper by Govil, etc., goes on to say “In reality, it is usually difficult to determine how long a fire has been burning before it is reported. [3]” This implies that civilian response is even slower than this. For the sake of having a mathematical number, 15 minutes will be used. Wildland firefighters therefore turn to prevention and probability of a forest fire. The most common measurements that is used in the field are probability of ignition (PIG) and fine

dead fuel moisture (FDFM). These measurements, which are normally performed by hand, or using a device such as a Kestrel Weather Station, allow the firefighters a better idea as to what the likelihood of a fire is in that area. Since many times resources are scarce during a fire, knowing when and where to focus efforts is a major contributing factor to containing a forest fire. These measurements, PIG and FDFM, are a standardized table that was developed by the National Wildfire Coordinating Group and Schroeder in 1969. Probability of ignition is estimated from the current temperature, shading from either forest canopy or cloud cover, and 1 hour fuel moisture content. This statistic is the probability that a fire will ignite in an area if a match is dropped on the ground, which is extremely useful for wildland firefighters to know. The table for PIG can be seen in Figure 1.

	DB Temp (°F)	1-hr Moisture Content (%)															
		2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0-10% shading	110+	100	100	90	80	70	60	50	40	40	30	30	30	20	20	20	10
	100-109	100	90	80	70	60	60	50	40	40	30	30	20	20	20	10	10
	90-99	100	90	80	70	60	50	50	40	30	30	30	20	20	20	10	10
	80-89	100	90	80	70	60	50	40	40	30	30	20	20	20	20	10	10
	70-79	100	80	70	60	60	50	40	40	30	30	20	20	20	10	10	10
	60-69	90	80	70	60	50	50	40	30	30	30	20	20	20	10	10	10
	50-59	90	80	70	60	50	40	40	30	30	20	20	20	10	10	10	10
	40-49	90	80	70	60	50	40	40	30	30	20	20	20	10	10	10	10
	30-39	90	70	60	60	50	40	40	30	30	20	20	20	10	10	10	10
	20-29	80	70	60	60	50	40	40	30	30	20	20	20	10	10	10	10
10-50% shading	110+	100	100	80	70	60	60	50	40	40	30	30	20	20	20	20	10
	100-109	100	90	80	70	60	50	50	40	40	30	30	20	20	20	10	10
	90-99	100	90	80	70	60	50	40	40	30	30	30	20	20	20	10	10
	80-89	100	90	80	70	60	50	40	40	30	30	20	20	20	10	10	10
	70-79	100	80	70	60	50	50	40	40	30	30	20	20	20	10	10	10
	60-69	90	80	70	60	50	50	40	30	30	20	20	20	20	10	10	10
	50-59	90	80	70	60	50	40	40	30	30	20	20	20	10	10	10	10
	40-49	90	80	70	60	50	40	40	30	30	20	20	20	10	10	10	10
	30-39	80	70	60	50	50	40	30	30	20	20	20	10	10	10	10	10
	20-29	80	70	60	50	50	40	30	30	20	20	20	10	10	10	10	10
60-90% shading	110+	100	90	80	70	60	50	50	40	40	30	30	20	20	20	10	10
	100-109	100	90	80	70	60	50	50	40	30	30	30	20	20	20	10	10
	90-99	100	80	80	70	60	50	40	40	30	30	20	20	20	10	10	10
	80-89	100	80	70	60	60	50	40	40	30	30	20	20	20	10	10	10
	70-79	90	80	70	60	50	50	40	30	30	30	20	20	20	10	10	10
	60-69	90	80	70	60	50	40	40	30	30	20	20	20	10	10	10	10
	50-59	90	80	70	60	50	40	40	30	30	20	20	20	10	10	10	10
	40-49	90	70	60	50	50	40	30	30	30	20	20	20	10	10	10	10
	30-39	80	70	60	50	50	40	30	30	20	20	20	10	10	10	10	10
	20-29	80	70	60	50	50	40	30	30	20	20	20	10	10	10	10	10
100% shading	110+	100	90	80	70	60	50	50	40	30	30	30	20	20	20	10	10
	100-109	100	90	80	70	60	50	40	40	30	30	20	20	20	20	10	10
	90-99	100	80	70	60	60	50	40	40	30	30	20	20	20	10	10	10
	80-89	90	80	70	60	60	50	40	30	30	30	20	20	20	10	10	10
	70-79	90	80	70	60	50	40	40	30	30	20	20	20	10	10	10	10
	60-69	90	80	70	60	50	40	40	30	30	20	20	20	10	10	10	10
	50-59	90	70	60	60	50	40	40	30	30	20	20	20	10	10	10	10
	40-49	80	70	60	50	50	40	30	30	20	20	20	10	10	10	10	10
	30-39	80	70	60	50	40	40	30	30	20	20	20	10	10	10	10	10
	20-29	80	70	60	50	40	40	30	30	20	20	20	10	10	10	10	10

Figure 1: Probability of Ignition Table [13]

Similarly, a table was developed for how moist the fuel of the fire is in a particular location. This measurement uses the current temperature and relative humidity in an area which will then yield the approximate humidity of the fuel. There are multiple tables depending on time of the year, time of day, and location, however a general reference fuel moisture table can be seen in Figure 2.

Dry Bulb Temp (° F)	Relative Humidity (%)																				
	0 to 4	5 to 9	10 to 14	15 to 19	20 to 24	25 to 29	30 to 34	35 to 39	40 to 44	45 to 49	50 to 54	55 to 59	60 to 64	65 to 69	70 to 74	75 to 79	80 to 84	85 to 89	90 to 94	95 to 99	100
10-29	1	2	2	3	4	5	5	6	7	8	8	8	9	9	10	11	12	12	13	13	14
30-49	1	2	2	3	4	5	5	6	7	7	7	8	9	9	10	10	11	12	13	13	13
50-69	1	2	2	3	4	5	5	6	6	7	7	8	8	9	9	10	11	12	12	12	13
70-89	1	1	2	2	3	4	5	5	6	7	7	8	8	8	9	10	10	11	12	12	13
90-109	1	1	2	2	3	4	4	5	6	7	7	8	8	8	9	10	10	11	12	12	13
109+	1	1	2	2	3	4	4	5	6	7	7	8	8	8	9	10	10	11	12	12	12

Figure 2: Fine Dead Fuel Moisture Table [14]

These two tables will be the leading factor in the device's ability to predict the probability of a wildfire. The settings regarding when the system will alert the user of hazardous conditions can be changed depending on the situation and environment where the unit is deployed.

This response time for the average fire can have dramatic consequences for the propagation of the fire. According to the Victoria Department of Environment, Land, Water, and Planning (previously Sustainability and Environment), forest fires “...can move as fast as 10.8 kilometers per hour (6.7 mph) in forests and 22 kilometers per hour (14 mph) in grasslands [5].” Presuming the slowest cited speed (6.7mph) and the optimal civilian response time of 15 minutes, the fire can travel 1.67 miles before emergency services are notified that there is a forest fire.

Another important detection scheme is those manned by professional fire detectors - fire towers and air fire patrols. Both have fallen out of favor but are still used in times of high fire

danger. Besides obvious limitations of these methods, such as human error, the most important is that these methods are not constantly employed - they are manned “for a period of 30 to 180 days depending on their work location and the season’s fire danger. [6]” While this is satisfactory, a 365-day remote surveillance system, or one that can be operated during the peaks of a fire season would catch more forest fires.

For this proposed system, are there any similarities and differences between the system and any current technologies? Yes, this proposal is focusing on a wireless environmental monitoring system implementing a long range and low powered communication protocol to achieve the task of sensing and relaying data back to the end user. One of the primary objectives of the system is to be able to transmit data over long distances where no pre-existing network infrastructure is available. Since this system would see deployment in a forest with limited networking capabilities, the communication would need to be handled entirely by each device or node. An existing communication protocol that would fit the criteria of being both long range, and low power, is the LoRa networking protocol. LoRa (Long Range) is a proprietary spread spectrum modulation technique by Semtech. It is a derivative of Chirp Spread Spectrum (CSS). [7] One such difference between most implementations of a LoRa system and the proposed system is the ability to form a mesh network from various LoRa nodes all communicating together. One such example of a typical application for a LoRa system is a city-wide meter reading collection system where devices send readings at very low frequency over a long distance to a data concentrator. This would be an example of a “one-hop” network, as each node is directly sending information back to a main processor in a star topology. [7] The difference in the proposed system is to use a mesh network, that is each system would be interconnected to each other therefore the system would not have to rely upon a single failure point of

communication back to the end user. For example, in a “one-hop” system, if a node were to be lost due to the extreme heat of a wildfire, all data coming from that location would be lost. If multiple nodes are scattered close to the front line, if one node is lost, other nodes would be able to continue to relay information back to the user. Another breakthrough technology that has recently been offered to the public is the use of the existing LoRa communications protocol, however, using a ground-based station with a satellite receiver. The same principles as a city-wide meter collection system would apply, however, instead of each meter communicating with another meter, the system would interface with low earth orbit satellites and the satellites would then offload collected data through a downlink station. Once the information is downloaded from the satellite it is provided to the user using a common node or internet connection. One such technology is from the company Swarm, that operates low earth orbit LoRa satellites for internet of things (IoT) applications. Currently this is the major difference between the proposed system and existing LoRa implementations. Since most of the other data collection will be done with standard sensors that are seen in most environmental monitoring systems, there can be similarities drawn between the two. For example, almost all commercial off the shelf (COTS) systems have sensors to measure temperature, humidity, and pressure. With other more specialized systems being able to measure precipitation amounts and snowfall totals.

Are there existing or patented technologies that may be relevant to the design? Upon researching various implementations of LoRa networks, large-scale off-grid systems setup in forests and large expanses of land have been developed. Research that was performed in Indonesia set up a multipoint LoRa network on the side of Mount Arjuno, which is a volcano in Indonesia, and the system was used to monitor coffee plantations over 1270.12 acres of land. To make it easier for farmers to know the condition based on microclimate, a data logging system



was needed that could monitor microclimate sensors such as air humidity and air temperature, soil moisture, soil temperature, and light intensity data by using the concept of the Internet of Things in Mount Arjuno region. [8] The final system that was implemented saw an average delay of 621.74 ms, and a packet loss of only 2%. [8] This research would provide relevance to the proposed system as not only did the packet loss remain low, which allows end users to make sure they are receiving correct information, but the system delay was very low for a long-range communication protocol. This means that if a node saw irregular conditions, it would only take on average 621.74 ms to notify the user, which would be much faster than traditional firefighting means such as a bystander notifying emergency services of a fire using 911. Another patented system that was submitted by Xidian University in China focused on an outdoor wireless communication system that was based on the LoRa protocol. This system performs communication without the need of a mobile network, has long transmission distance, has the interactive interface for information sharing, and can be applied to the fields, such as the Internet of Vehicles and the Internet of Things. [9] While this system does not specifically monitor the environment around it, it does have a GPS locating module, which allows for connection to global positioning satellites to accurately detect where the system is located on Earth. The proposed system would also utilize some type of global positioning, that way, if deploying nodes into remote locations, retrieving them would be made easier by use of satellite technology. Thus, this patented system would allow for reference of how a system like that would be designed. A second patented system that focuses on sending an SOS signal in an outdoor environment with very little infrastructure was submitted by the Jiangsu Maritime Institute in China. This module provides users a way to signal they are in distress while also providing the location of that user using GPS and a wireless transmitter module. [10] This patented technology will provide basic

understandings of how a critical system like an SOS transmitter implements other failsafe's such as the use of retransmitting data to make sure the important message of an SOS emergency is delivered to the user. The proposed system will need to make use of some sort of failsafe as messages of rapidly changing wildfire conditions must be delivered to the end user.

#### **1.4. Marketing Requirements**

[JCM, MS, SW, NS]

1. The device will measure environmental parameters that are indicative of a wildfire.
2. The device will be able to make accurate predictions of the probability of a wildfire based off environmental measurements.
3. The device should be able to wirelessly connect to a network regardless of location in the world.
4. The device location will be known to the user regardless of location in the world.
5. The device should notify users of the probability of a wildfire.
6. The device should use renewable energy to operate for an entire fire season.
7. The device should be easy to transport and deploy, regardless of environment, terrain, or pre-existing network infrastructure.

## **2. Engineering Analysis**

### **2.1. Circuits**

#### **2.1.1 Power Distribution**

[JCM]

To derive Engineering Requirement #1, several considerations were made regarding power distribution. Per E.R. #1, the device must be able to operate for a period of 6 months (an entire fire season) without being serviced. There are two main approaches that were considered to achieve this:

- A). Using a battery of such a capacity that could power the system for 6 months without recharging before needing to be replaced/recharged by manually interacting with the device.
- B.) Using a battery of smaller capacity that would be charged through some renewable resource (i.e., sunlight) where the power supplied by the renewable resource would outpace the power consumed by the rest of the system.

To analyze either of these options, the power draw of each of the subsystems must be ascertained to determine if option A or B are even feasible. The main subsystems of this project that consume power are the microcontroller, the communications module, and the environmental sensors. Based on preliminary research into possible component options, a baseline for power consumption was found for each of these systems.

First and foremost is the operating voltage of each subsystem must be discussed. The microcontroller, communications module, and each sensor potentially have different operating voltages. For example, the PIC series tends to operate on 3.3V where a few of the sensors were found to operate at 4-5V. This means that to use a single DC source, the voltage must be regulated down from the source voltage to each of these operating voltages.

Second is the current draw and when the device will be operating at that current. The microcontroller options that were considered tend to draw a current on the scale of micro-amps while operating. The communications unit tends to draw on the scale of milli-amps when sending and receiving data. The sensors' current draw varies depending on the type of sensors however a fair estimate would be on the range of milli-amps while the device is actively sensing. These are all low power draws when considered on their own, but when operating continuously, this consumption will quickly overtake a battery of reasonable size. To combat this, it was decided that only the microcontroller should be powered continuously. The microcontroller should then control the power flow to the other systems.

However, there are a few drawbacks to this approach. The common output voltage of the PIC series is 3.3V. This is not a high enough voltage to power some sensors and potentially the communication module, depending on the final choice. There is also the concern of drawing more current than the microcontroller can handle. To avoid this, it was decided that the communications unit and the sensors would be powered directly from the source, regulating the voltage down as needed, and the flow of that power would be restricted using a control system operated by the outputs of the micro-controller. It would suffice to think of this control system as a relay, where the contacts are actuated by the output of the microcontroller. A transistor will be more appropriate. This system will be implemented using a PMOS transistor, where the battery regulated battery voltage is applied to the source of the transistor and the sensor is connected to the drain. An output from the microcontroller will be connected to the base of the PMOS. This PMOS will be "logic level" meaning that a base voltage of 3.3 volts can allow higher voltages to pass from source to drain. Since the PMOS is "active low", when the microcontroller outputs a voltage, the drain voltage will be zero, otherwise drain will follow the source.

This allows the microcontroller to power the devices necessary for its current operation. For example, when the microcontroller sees fit to collect data, it may power the sensors. When it needs to send out data, it can power the communications module. This control, combined with programming, allows the device to save energy by only powering the devices it needs for as long as it needs them. After deliberation that can be found in section “2.2 Electronics” of this document, it was deemed necessary for two primary power modes. Under normal operation, the device will power the sensors and communication module every 15 minutes. The sensors will remain powered for 30 seconds and the communications module will remain powered for as long as it takes to send the data (roughly 3.7 seconds for example). The second power mode is a high-power mode where data is being collected and sent more frequently. In this mode the devices will be powered for the same duration, but powering will occur every 5 minutes.

Putting these ideas together, along with the preliminary research into possible components, a power consumption estimate can be calculated.

	Current draw[A]	voltage[V]	wattage[W]
wind speed sensor	0.0002	5	0.001
sunlight sensor	0.000045	3.3	0.0001485
temp sensor	0.0002	5	0.001
pressure sensor	0.000005	5	0.000025
humidity sensor	0.002	5	0.01
co2 sensor	0.02	5	0.1
microcontroller	0.000525	3.3	0.0017325
comms	0.85	3	2.55

Figure 3: Example Power Calculations

Figure 3 shows these example calculations for each component. However, Watts alone are not enough to choose a battery. Battery capacities are rated in either amp-hours or watt-hours, therefore time must be introduced into these calculations.

				Low power			High power		
	Current draw[A]	voltage[V]	wattage[W]	running time [hours]	Energy Consumed[Wh]	Energy Consumed[Wh]	running time [hours]	Energy Consumed[Wh]	Energy Consumed[Wh]
				per hour sample	per hour sample	per day	per hour sample	per hour sample	per day
wind speed sensor	0.0002	5	0.001	0.033333333	3.33333E-05	0.0008	0.1	0.0001	0.0024
sunlight sensor	0.000045	3.3	0.0001485	0.033333333	0.00000495	0.0001188	0.1	0.00001485	0.0003564
temp sensor	0.0002	5	0.001	0.033333333	3.33333E-05	0.0008	0.1	0.0001	0.0024
pressure sensor	0.000005	5	0.000025	0.033333333	8.33333E-07	0.00002	0.1	0.0000025	0.00006
humidity sensor	0.002	5	0.01	0.033333333	0.000333333	0.008	0.1	0.001	0.024
co2 sensor	0.02	5	0.1	0.033333333	0.003333333	0.08	0.1	0.01	0.24
microcontroller	0.000525	3.3	0.0017325	1	0.0017325	0.04158	1	0.0017325	0.04158
comms	0.85	3	2.55	0.001027778	0.0036	0.0864	0.004111111	0.0144	0.3456

Figure 4: Example Power Calculations Cont.

Figure 4 shows the power consumption in watt-hours of each component per hour and per day for both power modes.

The power consumption was calculated using Equation 1.

Eq. 1

$$E \left[ \frac{W \cdot h}{hour} \right] = V[V] * I[A] * t[s] * n \left[ \frac{iterations}{hour} \right] * 1/3600 \left[ \frac{h}{s} \right]$$

$E$  is the energy consumed by a device in a one-hour sample

$V$  is the operating voltage of the device

$I$  is the current draw of the device during operation

$t$  is the amount of time the device is powered for in a single iteration of receiving power

$n$  is the number of times that the device receives power in a one-hour sample

$1/3600$  is a conversion from seconds to hours, primarily used for  $t$ , to change the unit in the numerator from W-seconds (or Joules) to W-hours.

For practical purposes, the “High power” mode of operation was considered for the power consumption calculations. This calculation was carried out for each of the devices to determine the total power consumption in one hour of operation. This can then be multiplied by 24 [hours/day] to see the power consumption of one day of operation. This can of course be extended to one month and six months of operation. These are calculated by summing the values of the final column listed in Figure 4, multiplying them by 30 to estimate one month, then by taking this value and multiplying it by 6 to estimate 6 months. The final estimates are given in Table 1.

Table 1: Example Power Consumption

Operating Period	Total Power Consumption
1 month	19.69 W-h
6 months	118.2 W-h

This brings the topic back to the consideration of the two possible supply approaches:

- A.) Using a battery of such a capacity that could power the system for 6 months without recharging before needing to be replaced/recharged by manually interacting with the device.
- B.) Using a battery of smaller capacity that would be charged through some renewable resource (i.e., sunlight) where the power supplied by the renewable resource would outpace the power consumed by the rest of the system.

To achieve option A, a battery must be found that has a capacity of 117.7 W-h and operates at a voltage greater than or equal to the highest operating voltage of the system. In this case, that voltage is assumed to be 5 volts. This is possible; however, it is expensive. Another way to create this is to combine multiple batteries of smaller capacities in parallel, therefore increasing their effective capacity. This is also feasible but is also expensive as multiple identical batteries must be purchased. This issue of a battery's capacity proportionally increasing its cost can be remedied by considering option B. A battery with a capacity of 19.6 W-h and an operating voltage of 5 volts is far more common and cheaper.

Pursuing option B poses another problem. Option B only allows for 1 month of operation as opposed to the intended 6 months. This requires that the battery be charged regularly. Since this device is intended to be deployed in a remote environment, it must be charged using renewable energy that does not depend on existing infrastructure. Solar power was primarily considered for this project as it seemed the most appropriate. For the device to last for at least 6 months, a solar panel must produce energy at a rate that outpaces the energy consumed. Solar panels will produce their rated wattage during Peak Sun Hours (PSH).



The energy produced for one day of operation can be found using Equation 2:

Eq. 2

$$E \left[ \frac{W \cdot h}{day} \right] = P[W] * PSH \left[ \frac{h}{day} \right]$$

E is the amount of energy produced by the solar panel in one day's operation (at full sunlight)

P is the power rating of the solar panel

PSH is the Peak Sun Hours of the area that the solar panel is placed in

For example. If a solar panel is rated for 2 watts, and during the day has 4 PSH, then during that day the solar panel will produce 8 W-hours of energy.

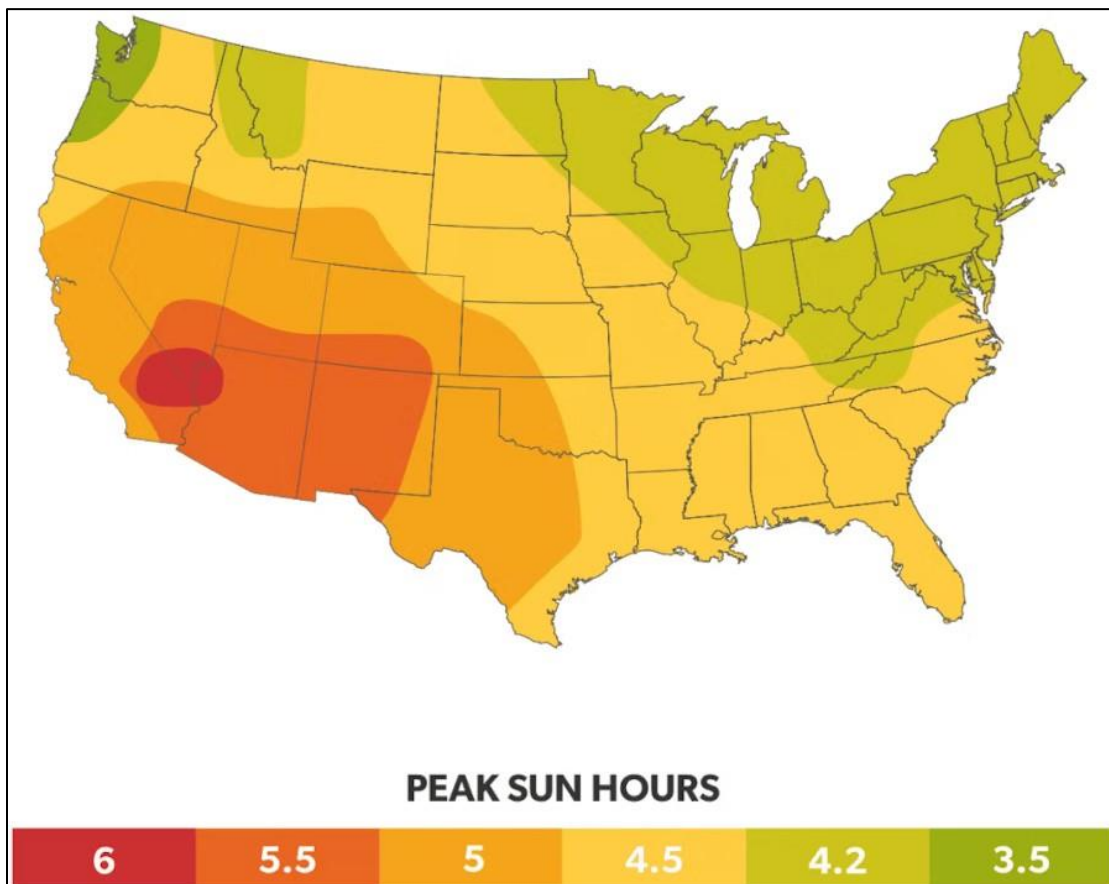


Figure 5: Peak Sun Hours Map [12]

The Peak Sun Hours of Ohio are 3.5 hours, however the PSH of the forest fire region are 5 hours [12]. 3.5 PSH will be used for device specifications as it is the lower value and a solar panel that can solve this design problem under those conditions will also be sufficient in the forest fire region. To replenish the 19.69 W-h battery, a solar panel must produce at least 19.69 W-h in one month. For example, if a 1-watt solar panel is considered, per equation 2, it only needs 19.69 PSH in an entire month to charge the battery. Since Ohio has on average 3.5 PSH per day, one month yields approximately 105 PSH. This signifies that a 1-watt solar panel will produce more than enough energy to charge the battery.

There are a few more considerations to be made regarding the solar panel. A solar panel tends to produce its rated voltage, provided any light. This voltage should be greater than or equal to the voltage of the battery and can be regulated if required. Since the solar panel is not producing a voltage if there is no light present, then it may act as a virtual ground and potentially drain the battery. To avoid this, a battery charging unit will be used. In the circuit shown in Figure 6, this battery charger is shown as a diode. Finally, the current produced by the solar panel changes based on the amount of light. The max current output of the solar panel should not exceed the batteries nominal charging current.



\*Note that they do not need to be equal, it would require another voltage regulation and another transistor for each different voltage

- $V_b = I_{sensor} = I_{s1} + I_{s2} + \dots + I_{sn}$
- $I_{load} = I_{sensor} + I_{comm} + I_{mc}$
- $I_{sol} \neq I_{load}$

These relations yield the following conclusions about the power supply system:

- Battery
  - Must be rated to supply a voltage that is greater than or equal to the highest operating voltage of the system
  - Must be rated to supply a current that is greater than or equal to the sum of the load currents
  - Must be rated at a capacity that is greater than or equal to one month's operation per the previous power consumption calculations
- Solar Panel
  - Must be rated to supply a voltage that is greater than or equal to the voltage of the battery
  - Must be rated to supply a current that does not exceed the charge current rating of the battery
  - Must be rated for a wattage that can supply enough energy during the peak sun hours of one month per the previous power consumption calculations
- Voltage Regulation
  - Each voltage regulator must be rated to accept a voltage and output a voltage depending on its position in the circuit

- Each regulator must be rated to accept a current that is greater than or equal to the current passing through it depending on its position in the circuit
  - for example, the voltage regulator furthest to the left in the diagram must be rated for an input voltage equal to that of the solar panel, an output voltage equal to that of the battery, and a current greater than or equal to the maximum output current of the solar panel
- Diode
  - The diode must be rated to operate at the voltage provided by solar panel after the voltage regulation
  - The diode must be rated to accept a current that is greater than or equal to the maximum output current of the solar panel
- Transistors
  - Each transistor must be rated to operate at the voltage provided after any voltage regulation, depending on its position in the circuit
  - Each transistor must be rated to accept a current that is greater than or equal to the maximum current draw relative to its position in the circuit
    - For example, the transistor furthest to the right must be rated for the sensor voltage, and for the sum of the individual sensor currents
- Fuses
  - Each fuse must be rated to operate at the voltage provided after any voltage regulation, depending on its position in the circuit
  - Each fuse must be rated to prevent any current that is 25% more than the maximum current drawn relative to its position in the circuit

The microcontroller, communications module, and sensors should be chosen to operate at a minimal current draw and at similar voltages. This will limit power consumption and limit the number of voltage regulations. Ultimately, the power circuit will need to be modified to supply the components, regardless of the components meeting these considerations.

## **2.2. Electronics**

### **2.2.1 Microcontroller**

[NS, MS]

Per Engineering Requirement #3, the device should accurately trigger sensor collection while utilizing low power modes and interrupts. To ensure that the system minimizes power draw, it should only be operating when it needs to. This can be done using hardware interrupts that are built into most microcontrollers. In the PIC microcontroller family sold by Microchip, a timer interrupt can be used either by using an internal or external clock. The most common internal clock is an 8 MHz crystal, however, for a system where the measurement intervals could be up to 15 minutes, a clock that is slower will be needed to keep accurate timing. This system will operate in two separate modes dictated by environmental conditions. In high data rate mode sensors will be polled every 5 minutes. This is activated when conditions yield favorable to a forest fire and more finite data is required. For analysis, an arbitrary PIC microcontroller with a Timer1 peripheral will be used. This Timer1 will be connected to an external 32.768 kHz clock crystal to provide more improved "real time clock" features than the more standard and faster 8 MHz clock. The slower clock allows for pre-scaling that is more favorable for timing events in the "seconds" and "minutes" instead of "milliseconds". Using a timer allows for the microcontroller to remain in a lower power mode, and then be triggered by the timer interrupt to

"wake up" and perform some function. If a 32.768 kHz clock is used and connected to Timer1, the pre-scaler can be set to 1:256. After scaling the input clock signal using:

$$\frac{32,768 \text{ Hz}}{256} = 128 \text{ Hz}$$

This 128 Hz signal will trigger a register in the PIC microcontroller, setting a register called the PR1 register, or periodic reset register to 128 – 1 will yield a clock event or interrupt every 1 second, or 1 Hz. At the 1 second interval, a condition will be checked to see if it matches the required time. In the case of every 5 minutes, for a high data rate mode, the timer would need to trigger 300 times this is derived by  $5 \text{ minutes} \times 60 \text{ seconds}$ , and on that 300th time, the sensor data would be read. During all other times, the microcontroller would be idle, sensors would be turned off, and data would not be collected.

Another option for this system is low data rate mode. This is when the sensors on the system are triggered every 15 minutes. A mode like this can be useful for extremely low power applications, or if the environment is not probable to a wildfire, so increased data is not needed. Using the same setup as before, the clock would trigger every 900 times, derived from  $15 \text{ minutes} \times 60 \text{ seconds}$ , and on that 900th time, the sensor data would be read along with a packet that would be sent to the communication module for wireless transmission.

Therefore, from Engineering Requirement #3, it is necessary to have a 32.768 kHz external clock with a hardware pre-scaler of 1:256, or a microcontroller that supports a 16-bit timing register, in which the hardware pre-scaler could be omitted.

### 2.2.2 Sensors

[MS]

To accurately sense if a fire is in the vicinity of the device multiple sensors must be used. To fuel the PIG and FDFM calculations with measurements and data, at least three different types of measurement are needed: sunlight, temperature, and humidity. Additional factors could be useful: wind speed to tell the potential ferocity of a fire and a barometric pressure sensor, in aiding to prevent a forest fire before it even occurs.

For the wind speed sensor, a linear analog voltage is the output. This is performed by 3 wind cups on the sensor. When air is blown into them, a rotor cup is rotated in the body of the sensor to output a voltage signal. To interpret this on the side of the microcontroller, an ADC (Analog-to-Digital Converter) is used. The analog voltage is taken, sampled, and applied to a range of 0-3.3V digitally. For the PIC24, this voltage is assigned a range linearly along an index of 0-1023. (1024, or 10 bits, total different assignments, to correspond to 0-3.3V analog). This is then translated, through equations in software, to an approximation of the analog voltage as a digital value. This value is stored in an ADC buffer and passed to an array to store for transfer with the SWARM antenna module. This value is not used in any calculations but is useful for determining potential strength of fire for a team looking to approach the incident.

All other sensors will use a serial communication interface to interact with the microcontroller. All other sensors will be utilizing the I2C protocol. (see Section 2.4). The three sensors required for FDFM and PIG calculations are temperature, humidity, ambient light (for sunlight, or shading). An extra sensor that is also used in predicting the potential conditions for a forest fire will be used: barometric pressure.



An integrated-circuit (IC) temperature sensor will be used for this application. The one being used here specifically is a silicon bandgap temperature sensor. This sensor takes advantage of the temperature characteristics of a BJT. The base-emitter voltage ( $V_{be}$ ) is related to temperature. The simplified form of this equation is shown in the equation below:

$$V_{be} = \frac{kT}{q} \cdot \ln\left(\frac{I_{c1}}{I_{c2}}\right)$$

$k$  is Boltzmann's constant,  $q$  is the charge of an electron, and, most importantly,  $T$  is temperature in Kelvin. This relation can be used to find the temperature of the surrounding area using a silicon IC. This temperature will be used in both the FDFM and PIG equations (for these equations/tables, see Section 1.3: Background).

Included in the same package as the IC temperature sensor, but as a different chip, is a humidity sensor, used for fine-dead-fuel-moisture calculations (see Figure 2). The humidity sensor used is a capacitive humidity sensor. The sensing method is quite simple: a strip of metal is placed between two conductors. The strip of metal is so designed as to have differing electron (and therefore, electrical) capacities under different humidity levels. This allows for an accurate and sensitive humidity measurement. This is used in the FDFM calculation to measure the relative humidity of the trees, our "fuel."

The last sensor required to perform our calculations is "shading," or sunlight. Sunlight is measured in lumens – the quantity of light emitted. When speaking of luminance over a certain area, illuminance, or lumens per meter squared, is used in SI units. This unit is called the "lux," and will be used for shading in our calculations, as it is the unit measured via our light sensor. Light sensors use a photoreceptor to convert light to electricity.

The shading parameters in both PIG and FDFM tables are binary in nature – either “sunny” (51% or higher compared to direct sunlight) or “cloudy” (50% or lower compared to direct sunlight). A table showing a relationship between relative outside conditions and the illumination level is shown in Figure 7.

Condition	Illumination	
	(ftcd)	(lux)
Sunlight	10000	107527
Full Daylight	1000	10752
Overcast Day	100	1075
Very Dark Day	10	107
Twilight	1	10.8
Deep Twilight	0.1	1.08
Full Moon	0.01	0.108
Quarter Moon	0.001	0.0108
Starlight	0.0001	0.0011
Overcast Night	0.00001	0.0001

Figure 7: Illumination level to condition [17]

Using this, a relationship to be used for light sensing can be used. While not perfect, this table can be used to have a cutoff for lux values for cloudy and sunny conditions. Using the “full daylight” value, 10752, and dividing it by two, should be sufficient. This will tell us which part of both the PIG and FDFM table we need to refer to when making calculations.

The last sensor used will not be used in the PIG or FDFM calculation but is important in determining good conditions for a wildfire: barometric pressure. A barometric pressure sensor relies on a small metal box on the system, made from copper and another metal alloy, that flexes in the face of different atmospheric conditions. This can be used to determine barometric pressure. This is measured in mm Hg, or millimeters of Mercury.

### 2.3. Mechanical

[NS]

Engineering Requirements #7 states that this device should be lightweight, less than 3kg. This is desirable to end users as the device will be and easily man portable for deployment in forest, mountainous, or adverse conditions, where the primary means of getting to these locations would likely be hiking, or on foot. As the system will be exposed to the elements for at minimum a 6-month period without user intervention, it is also imperative that the system is resistant to outside elements, dust, dirt, and debris. Breaking down the mechanical drawing that can be found in Figure 18 and Figure 19, the main sources of weight for this system are the solar panel, enclosure, batteries, sensors, external antennas, wiring, connectors, and finally fasteners. Table 2 provides a breakdown as to each of these components and the probable weights that are associated with each. These weights are subject to revision as final part selection and availability will further dictate the final weight.

Table 2: Estimated Weight of Major Components

Component	Estimated Weight
Solar Panel (5W ETFE panel for consideration)	188 Grams
Enclosure (Waterproof Hinged Electrical Junction Box)	1.28 Kilograms
Batteries (Three 18650 Style Battery Cells)	136.95 Grams
Temperature Sensor	25 Grams
Sunlight Sensor	15 Grams
Pressure Sensor	10 Grams
Relative Humidity Sensor	15 Grams
Wind Sensor	111.2 Grams
RF Antennas (GPS and Data)	131.2
Wiring (5% of Total Weight)	95.61 Grams
Connectors (5% of Total Weight)	95.61 Grams
Fasteners (15% of Total Weight)	286.85 Grams

From Table 2 the estimated total weight of the device will be 2294.82 grams, or 2.3 kg, which is under the 3 kg weight limit that is dictated from Engineering Requirement #7. This weight is not finalized, as final component selection has not been established, however, this is the theoretical weight that should be expected in the design.

## 2.4. Communications

### 2.4.1 Wired Communications

[NS]

Engineering Requirement #4 states the data collected from the device should be limited to less than or equal to 192 bytes worth of data to ensure low power and bandwidth wireless transmission. The formatting of the communications packet that will be filled with sensor data is as follows:

Communications Packet Example:

```
{LN:X99.9999, LT:X99.9999, ALT:9999.9, PIG:999, FDFM:99, SUNIR:9.99,  
SUNUV:99, TEMP:999, BARPRESS:99.9, RELHUMID:99.9, WINDSPD:999,  
WINDDIR:999, SOILMOIST:99.9, BATT:999}
```

The “9” following each field is placeholders for actual sensor data or calculations the microcontroller makes, but this format represents the absolute maximums the system would see. To make sure that Engineering Requirement #4 is met, ASCII formatting will be used to represent the letters that make up the communication packet. Each symbol in this packet will be represented by an ASCII character. An ASCII character has a size of one byte. After accounting for each character, there would be at most 179 total characters in a message, depending on the current readings from the sensors resulting in a total size of 179-bytes. Therefore, it is theoretically possible to package all necessary data for wildfire and environmental monitoring into a communication packet that is 179-bytes. The 192-byte limit is imposed as an engineering requirement due to the wireless communication modes that are in consideration. The satellite

communication technology provided by Swarm, limits the user to 192-byte messages, therefore it was imperative to make sure that all the data needed to be transmitted was within this limit.

Regarding sensor communication within the system, the I2C protocol is the main standard that will be followed. Other communication protocols like SPI and UART for the sensors were considered, however, since this system employs many sensors, the addressing of I2C that only requires a data and clock line allows for a much better implementation than other protocols that would need a select line for each device. This would add more complexity in not only the coding but the layout of the final system.

The satellite communication module that is provided to a designer by Swarm requires a UART interface to address the module. Although this module is not the final communication system this device will be using, selecting a microcontroller which can support this is imperative. Lastly, per Engineering Requirement #2, data from the sensors should be able to be stored in an offline format on the device. This could be accomplished using an SD card. If an SD card is used an SPI peripheral on the microcontroller would need to be used to interface with this device.

An example of the I2C communication protocol sequence is provided in Figure 8, these 8-bit messages are used to communicate with the temperature sensor specifically but is similar for all I2C sensors.

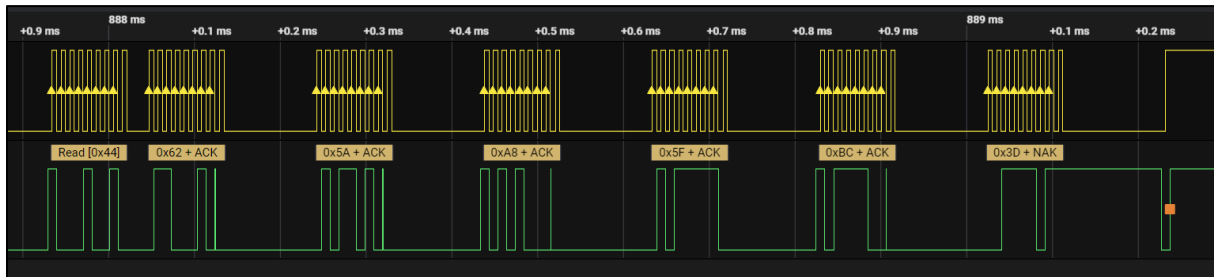


Figure 8: Example I2C Communication to SHT31 Temperature Sensor

The messages that are being sent in this string of communication is to read temperature and humidity data. This is done by first sending the device address, which for the SHT31 weather-proof temperature and humidity sensor, is 0x44. Next is to read in the three 8-bit packets that represent the 32-bits worth of temperature data followed by the two 8-bit packets that represent the humidity data. On the last read of humidity data, a no acknowledgment from the master device must be sent to end communication with the device. The code for implementing this using a PIC24FJ128GA010 can be seen in Table 17 in the appendix

For demonstration, the Explorer 16/32 development board was used in combination with the PICFJ128GA010 microcontroller. This allowed for the data being read in from each sensor to be displayed on the built-in LCD. In Figure 9, the temperature and humidity of the room is shown, and in Figure 10, the pressure and humidity are shown. Displaying this data on the LCD allows for a quick and easy way to make sure that the sensors are reading and outputting the correct data. All data also must be converted from a binary representation to a human readable version, the supporting code to perform these tasks can be found in the appendix.

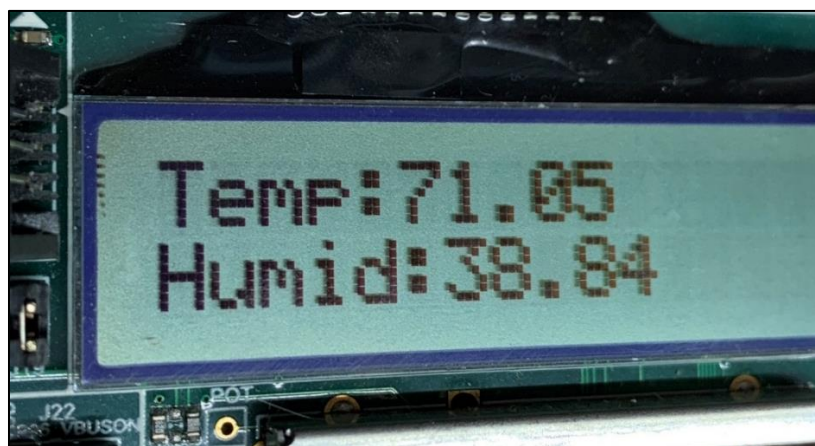


Figure 9: Displaying SHT31 Temperature and Humidity Data

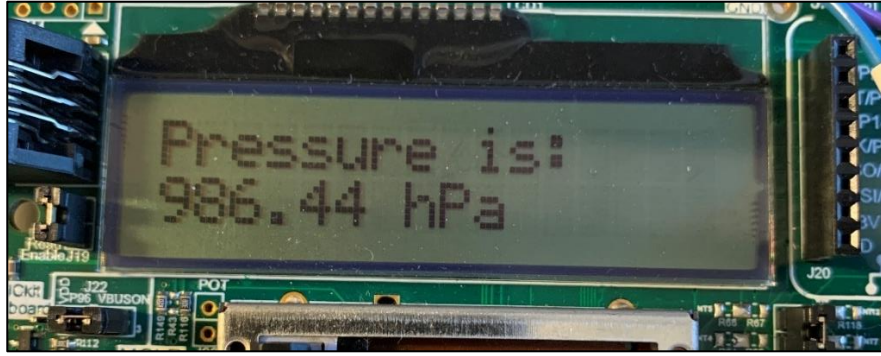


Figure 10: Displaying LPS33HW Pressure Sensor Data

For the final design, each of the communication protocols, I2C, UART, and SPI will need to be implemented to allow for seamless recording and transmission of current environmental characteristics. A short example of what this communication flow will look like is provided in Figure 11, this capture from the Saleae Logic Analyzer Logic 2 Software shows multiple communication protocols all being used simultaneously. The yellow text is data being sent through UART out of the Swarm M138 modem. The green text is UART data being sent from the PICFJ128GA010 microcontroller to the M138 modem, and the blue text is I2C SDA (Data) and gives a synopsis of what I2C data is being read or written at that time. Therefore, to summarize, the M138 modem will startup and initialize, checking current firmware and serial number. Next the \$M138 DATETIME\*56 command is issued. This is in a NMEA standard formatting with the '\$' being following by a command, and the '\*' is followed by a NMEA checksum in hexadecimal. The checksum code that is implemented is found in Table 18 in the appendix. After the \$M138 POSITION\*4e command is issued stating there is GPS lock, the PIC24 issues the command to send over the pressure data. The pressure data is read from the sensor and formatted into the string: "Pressure from ASEC 509 is 752.03 hg". In the command, the AI stands for the application ID, which is user defined, and the HD is the hold duration, or how long the message should be held on the M138 modem if it is not sent, with the maximum



time being 13 months. Lastly, the M138 modem sends an OK command with the message ID which will match the message in the Swarm Hive dashboard.

```
\x1B[0m
Swarm MPCIE (MPCIE) Bootloader
Copyright (c) 2019-22 Swarm Technologies, Inc
Version: 2022-05-16T21:46:13,v2.0.0
.....
Checking image MD5 in STM32 flash... valid
Identifying SPI flash... FOUND (Mfr 0x9d ISSI Type 0x6017)
Checking image MD5 in SPI flash... INVALID
Transferring control to application...

\x1B[0m$M138 BOOT,POWERON,Swarm M138 (M138)*44
$M138 BOOT,POWERON,Copyright (c) 2019-22 Swarm Technologies, Inc*2d
$M138 BOOT,POWERON,LPWR=n,WWDG=n,IWDG=n,SFT=n,BOR=Y,PIN=Y,OBL=n*43
$M138 BOOT,VERSION,2022-05-16T21:45:08,v2.0.2*2f
$M138 BOOT,DEVICEID,DI=0x003f3e*2f
$M138 BOOT,RUNNING*2a
$M138 DATETIME*56
$DT 20221109190445,V*4c
$M138 POSITION*4e
$GP 10*36
$TD AI=40,HD=7200,"Pressure from ASEC 509 is 752.03 hg"*26
write to 0x5D ack data: 0x11 0x10
write to 0x5D ack data: 0x28
read to 0x5D ack data: 0x08 0x6D 0x2F
$GP OK*33
$TD OK,4669679894530*2e
█
```

Figure 11: Example Communication Flow

Therefore, when considering a microcontroller to select, it is crucial that the device has at least one I2C, one UART, and one SPI peripherals to use to interface with external devices.

## 2.4.2 Wireless Communications

[NS]

Market Requirement #3 states that the device should be able to wirelessly connect to a network regardless of location in the world. This engineering requirement very quickly narrows down the different systems that could possibly be used. At first cellular may be considered, however, if there are no pre-existing cellular towers in range for the system to use, then

communication from the device is no existent. The next system that was considered was some type of hopping LoRa network that would use each sensor to communicate with the next so a “chain” of sensors could be established to talk to the furthest node. The limitations of this are that a sensor could not be dropped anywhere in the world. A small infrastructure of sensors would need to be established so that the user could talk to each device. When only one sensor is needing to be deployed this creates a large overhead that may not be desirable to users.

A new technology that is highly favorable for this system would be LoRa satellite communication. This uses the same chirp spread spectrum technology that makes LoRa so powerful for long range communication, but with the added benefit that a system could be placed anywhere in the world, and sense the satellites are low earth orbit, meaning that they are orbiting the Earth, they have global coverage throughout the day. One such technology that is being considered for this system is the company Swarm’s M138 satellite modem. This sub-1.5-inch mini PCIE sized modem will allow for easy integration into a small device like this weather monitoring station. Since Swarm has global coverage, this would satisfy Market Requirement #3 that the device should be able to wirelessly connect to a network regardless of location in the world.

However, there are limitations to this approach, since the satellites are low earth orbit, transmission is not instantaneous. The device will have to wait until a satellite is passing overhead to upload the data to it. According to Swarm’s Satellite Pass Checker [15], the University of Akron sees satellite passes from Swarm “cube-sats” anywhere from 5 minute to 20-minute intervals. Figure 12 shows a cube-sat, with dimensions of only 11 x 11 x 2.8 cm. Since these satellites are extremely small, they are much more cost effective to launch into space. The current Swarm constellation of satellites consists of 150.

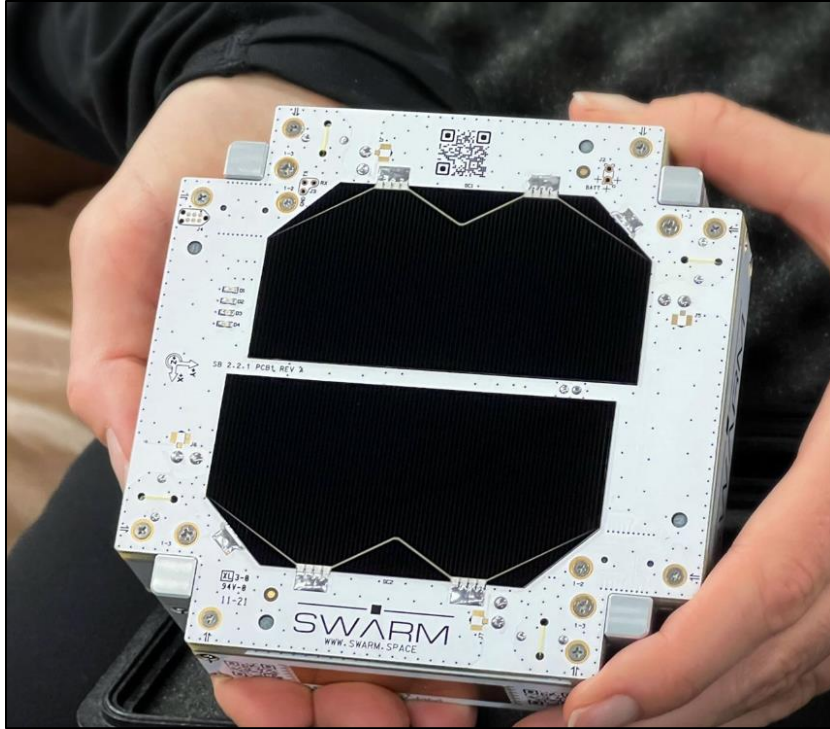


Figure 12: Swarm “Cube-Sat” Satellites

During testing and evaluation of the M138 Modem, it was seen that total turnaround time, from when the message was uploaded to a passing satellite, to when that data was available on the Swarm dashboard was only 2-5 minutes on average. This is significantly faster than what was expected and allows for much quicker retrieval of data that is recorded in the field.

To communicate with these satellites a  $\frac{1}{4}$  wave antenna tuned to 137 – 150 MHz was used with a supporting ground plane. For testing, a 50 cm long 20 AWG wire was used as a ground plane, however, the solar panel included in this design will be an adequate replacement when finalized. Figure 13 shows the implementation that was used for testing. This 3D printed part in green mounts the data antenna and the GPS antenna on a piece of steel stock away from the building for better receive signal strength to and from the antenna.

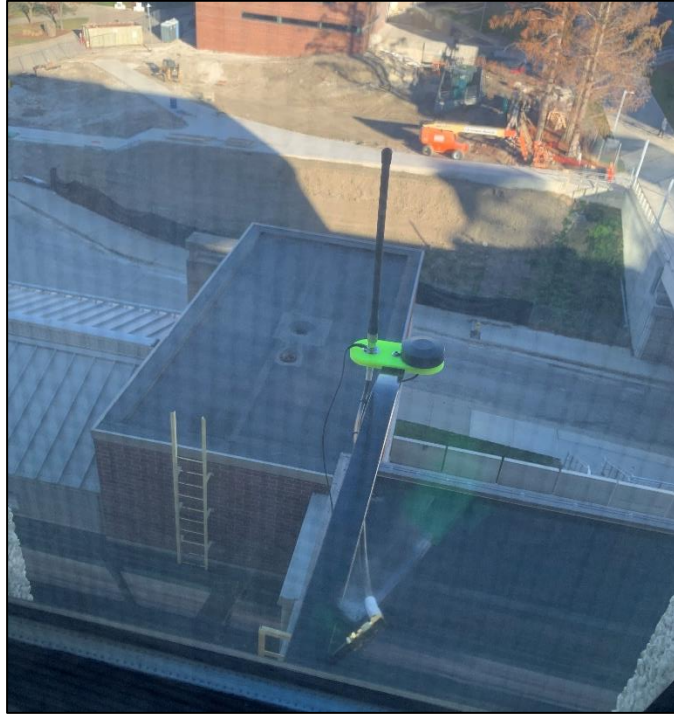


Figure 13: Antenna Implementation for Testing

Since this antenna was mounted at a position where roughly 50% of the sky was obscured by the building, the satellite pass durations were also limited to 50% of its actual time. However, there did not appear to be any issue in sending data, as 25 test messages with various amounts of data were all sent successfully and displayed on the Swarm dashboard along with the GPS location of the sensor. In the field, as the oscillation on the GPS receiver stabilizes it is expected to provide even more accurate timing and position data as starting and stopping the oscillation does not allow for adequate time to stabilize, but this was all during a testing phase.

## 2.6. Computer Networks and Software

[NS]

From Engineering Requirement #6 the device should present environmental data and probability of wildfire in an easy to interact with user interface. Building an application such as Figure X will allow the user to monitor easily and accurately each of the sensors in their network. Creating this application will provide a location where all data can be accessed and managed, along with providing an appealing user experience. This application can make use of open-source mapping applications such as OpenStreetMap along with Node.JS to make an easy-to-use user interface. These resources could then be hosted on either a client ran server, or a cloud-based web hosting system to allow users to access their data anywhere in the world that has internet connection. One such wireless satellite communications provider is Swarm that operates Low Earth Orbit satellites. If this system is used, once data is sent to the satellite, it is then offloaded and to a ground station. This ground station will then upload data to the Swarm “Hive” which allows subscribers of this system to access the data using either a REST API, or through web hooks to pull and consume that data into other applications. This will make data management and presentation easy using a custom-built application that pulls data from the Swarm Hive.

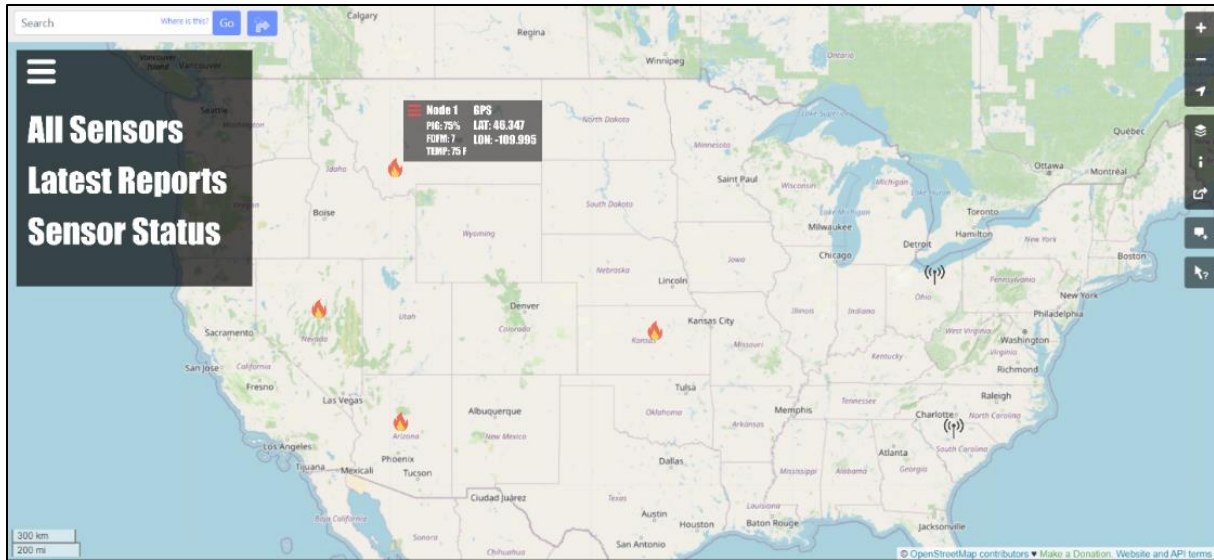


Figure 14: User Interface Example

[SW]

According to Engineering Requirement #6, the device should be able to calculate the probability of wildfire occurrences based on measured environmental factors. Given the most recently recorded data, it should be able compare its measurements to those that would indicate a conduciveness for the creation or prolonging of a wildfire. This would be achieved by creating a program that utilizes the measurement thresholds from the tables for PIG and FDFM (previously shown in Figures 1 and 2 respectively). Ideally, this would be done by parsing a text file with an equivalent format as the communication packet mentioned in the Wireless Communication section. If the conditions being actively recorded by the device are near, or exceed, the limits of what is considered acceptable, then a warning would be sent to an end user in addition to the data that is normally sent. Pseudo Code Representing the functionality of the software is shown in the following table.

Table 3: Fine Dead Fuel Moisture and Probability of Ignition Calculator Pseudocode

The goal of the program is to parse a text file for relevant information and then calculate values for probability of ignition and Fine Dead Fuel Moisture

```

class weatherData {
    Declare variables for temperature, wind speed, altitude, humidity, etc
    etc...
    Declare functions used to assist in later computations relative to class
    variables
}

function valueExtractor(Argument 1 ) {...}
function valueAuthenticator(Argument 1) {...}
function calculatePIG(Arguement 1) {...}
function calculateFDFM(Argument 1){...}

{
    In the main function

    Declare class variable.

    Pass class variable by reference when calling function "valueExtractor".
    Extracts relevant information from a text file and inserts their values into
    their perspective class variables. Will report an error if failed.

    Pass class variable by reference when calling function "valueAuthenticator".
    Authenticates values to make sure they are within the limits designated by
    programmer (no negative values where they are not applicable and setting a
    general upper limit). Outputs values and gives notice to errors if they arise.

    Pass class variable by reference when calling function "calculate FDFM".
    Output
    calculation result.

    Pass class variable by reference when calling function "calculate PIG".
    Output calculation result.
}

```

The following figure shows examples of the software's output when given randomly generated measurement values and their resulting FDFM values.

```
Test Case #1:
boolb@LAPTOP-AET80J50 MINGW64 ~/Documents/SDP_PIG_FDFM_
$ ./a
Altitude is 999.9 m
Sun irradiance is 9.99 W/(m^2)
Sun UV is 99 lumen
Temperature is 21°C or 69.8°F
Value out of range. Barometric pressure cannot exceed 99.9 Hg
Relative Humidity level is 38.84%
Wind Speed is 68mph
Wind Direction is North,(19°)
Battery is 60.9%

The Fine Dead Fuel Moisture Rating is 8
(Note: Based on a scale that ranges from 1 to 14; 1 representing conditions are
at their most conducive
to the start/prolonging of a wild fire and 14 being the least)

Test Case #2:
boolb@LAPTOP-AET80J50 MINGW64 ~/Documents/SDP_PIG_FDFM_
$ ./a
Altitude is 99.9 m
Sun irradiance is 9.99 W/(m^2)
Sun UV is 99 lumen
Temperature is 81°C or 177.8°F
Value out of range. Barometric pressure cannot exceed 99.9 Hg
Relative Humidity level is 78.84%
Wind Speed is 64mph
Wind Direction is North east,(49°)
Battery is 90.9%

The Fine Dead Fuel Moisture Rating is 10
(Note: Based on a scale that ranges from 1 to 14; 1 representing conditions are at their most conducive
to the start/prolonging of a wild fire and 14 being the least)
```

Figure 15: Example of FDFM Calculation Tests



## 2.7. Embedded Systems

### 2.7.1 External Storage Module

[NS]

Engineering Requirement #2 states the device should be able to store time stamped data from all sensors for at least a typical 6-month fire season in case wireless communication windows are missed, or data needs to be reviewed after the season. Using the same format as the communication packet except adding timestamp data the total size of each sensor capture would be 212-bytes. Since there are two modes in this system, one is a high data recording mode that triggers every 5 minutes, and a low "idle" mode that triggers every 15 minutes. Taking a worst-case example where the system is in high data recording mode for an entire day, that would produce:

$\frac{1440 \text{ minutes}}{5 \text{ minute}} = 288 \text{ total data captures per day}$
$212 \text{ bytes} \times 288 \text{ captures} = 61,056 \text{ bytes per day}$
$61,056 \text{ bytes} \times 182.5 \text{ days in a typical fire season} = 11,142,720 \text{ bytes or } 11.2 \text{ MB}$

From the calculations performed above, a storage device that is larger than 12 MB would be sufficient to store the data recorded from the wildfire season onto the system for later review. A module like an SD card could be utilized in this scenario because it can be interfaced through SPI using a PIC microcontroller.

### 2.7.2 Global Positioning Module

[NS]

Engineering Requirement #7 needs the devices' global positioning ability to be at least 95% accurate. Using an industry standard GPS module, such as a low cost UBlox module, which has the benefit of the standards associated with global positioning data within the United States.

The United States government is committed to providing GPS at the accuracy levels specified in the GPS Standard Positioning Service (SPS) Performance Standard. The accuracy commitments do not apply to GPS devices, but to the signals transmitted in space. For example, the government commits to broadcasting the GPS signal in space with a daily global average user range error (URE) of  $\leq 2.0$  m (6.6 ft.), with 95% probability, across all healthy satellites in constellation slots. Actual performance is typically much better. On April 20, 2021, the global average URE across all satellites was  $\leq 0.643$  m (2.1 ft.), 95% of the time. [16]

Therefore, it is feasible to expect that this system can provide accurate GPS data, within 2.1 ft, 95% of the time when uninterrupted or non-degraded signals are received using a commercial off the shelf (COTS) style GPS receiver.

The Swarm M138 modem has the luxury of a built in GPS receiver module. This system is a uBLOX GPS module that provides the users with latitude, longitude, and altitude which can all be queried using serial commands sent to the modem. The GPS module uses U.FL connectors to connect an external GPS antenna to it. The antenna that was used for testing and demonstration was a YIC ATGG46015 GPS and GLONASS antenna. This will allow for the reception of both GPS and GLONASS signals which will provide complete global coverage to the system.

### 3. Engineering Requirements Specification

Table 4: Engineering Requirements

[JCM, MS, SW, NS]

Marketing Requirement	Engineering Requirements	Justification
6	1. The device will be self-powered and will harvest a minimum of 656 milliwatt-hours of energy per day for the entire fire season.	The device will last for an entire fire season without needing to be serviced.
1,3	2. The device should be able to store 61 kilobytes of time stamped data from all sensors per day for the entire fire season.	Storing the data allows recovery in case wireless communication windows are missed, or data needs to be reviewed after the season.
1,2,6	3. The device should accurately trigger sensor collection every 15 minutes under normal operations and every 5 minutes when fire conditions are detected.	This allows enough data to be collected to predict/detect a fire and lowers power consumption
3,4,6	4. The data collected from the device should be limited to less than or equal to 192 bytes worth of data for each message transmission.	This ensures low power consumption and low bandwidth wireless transmission.
3,4	5. The device's global positioning ability should be within 2 meters of its actual position, with 95% accuracy.	The device will be able to be in low visibility environments.
2,3,4,5	6. The system will measure and display the current environmental temperature, relative humidity, barometric pressure, wind speed, and sunlight and calculate the probability of wildfire.	The interface will allow a user to clearly understand the state of the environment, as well as the device, and act accordingly.
7	7. The device should be lightweight, less than 3 kilograms.	The device should be easily man portable for deployment in forest, mountainous, or adverse conditions.

1, 4	8. The system will measure and record the temperature of the environment accurately to 0.1 degrees Fahrenheit.	For an accurate temperature reading of the wildfire area, and to ensure the calculations used to show wildfire probability are accurate. Fahrenheit is the temperature measurement used in the United States.
1, 4	9. The system will measure and record the relative humidity of the environment accurately to 0.1%.	For an accurate humidity reading of the wildfire area, and to ensure the calculations used to show wildfire probability are accurate. Relative humidity is measured in percentage.
1, 4	10. The system will measure and record the barometric pressure of the environment accurately to 0.1 mmHg.	For an accurate humidity reading of the wildfire area, and to ensure the calculations used to show wildfire probability are accurate. Barometric pressure is measured in mm Hg.

#### Marketing Requirements:

1. The device will measure environmental parameters that are indicative of a wildfire.
2. The device will be able to make accurate predictions of the probability of a wildfire based off environmental measurements.
3. The device should be able to wirelessly connect to a network regardless of location in the world.
4. The device location will be known to the user regardless of location in the world.
5. The device should notify users of the probability of a wildfire.
6. The device should use renewable energy to operate for an entire fire season.
7. The device should be easy to transport and deploy, regardless of environment, terrain, or pre-existing network infrastructure.

## 4. Engineering Standards Specification

Table 5: Engineering Standards Implemented

[JCM, MS, SW, NS]

Category	Standard	Use
Safety	IEC 62133, UL 2054, UN/DOT 38.3	These safety standards correlate directly to the safe use of LIPO and/or Lithium batteries, which will be a primary design consideration.
Communications	I2C, UART, LoRa, Chirp Spread Spectrum (CSS), NEMA	All these communication protocols will be used to interact with sensors, communication modules, and wireless transmission.
Data Formats	JSON, ASCII	The formatting of sensor data which will be displayed on the devices web application will make use of JSON formatting.

Design Methods	3D Printing, Solidworks CAD, KiCAD PCB Design, Rapid Prototyping	Various modern computer aided design packages will be used to produce this system allowing for the reduction of cost and improvement of ease of manufacturing.
Programming Languages	C/C++, Python, JavaScript, HTML, CSS, React.js	The microcontroller will primarily be coded in C/C++ while the web application will use a mixture of Python, JavaScript, HTML, CSS, and React.js.
Connector Standards	RJ45, Molex, JST, SMA, U. FL	RJ45 connectors will be used to interface and program the microcontroller, while Molex and JST connectors will primarily serve as battery and sensor connectors. SMA and U. FL RF connectors will be used for GPS and Data antennas on the system.

## 5. Accepted Technical Design

### 5.1. Hardware Design

#### Hardware - Level 0

[JCM, MS, SW, NS]

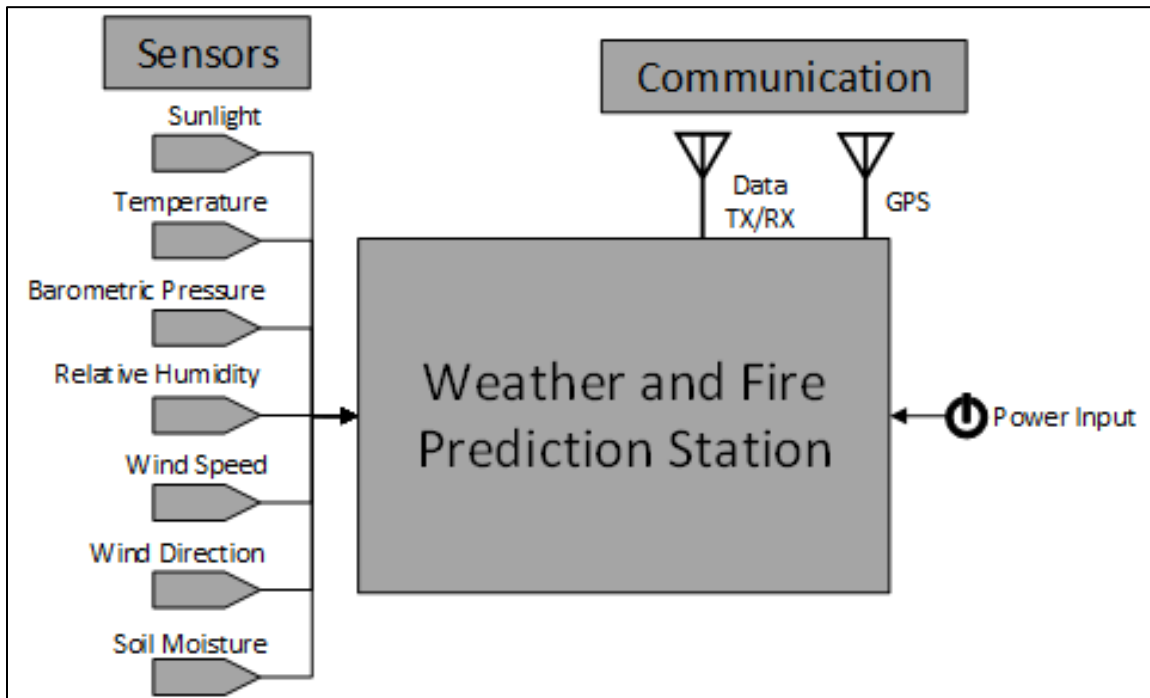


Figure 16: Level 0 Hardware Block Diagram

Table 6: Functional Requirement Table for the Level 0 Hardware Block Diagram

Module	Weather Monitor Station – Overview
Designers	DPS Team 11 (JCM, NS, MS, SW)
Inputs	<ul style="list-style-type: none"> <li>- Power</li> <li>- Environmental data (temperature, relative humidity, wind, air pressure, soil moisture)</li> <li>- GPS tracking (accurate to ~10ft)</li> </ul>

Outputs	- Environmental data
Functionality	<p>Takes in environmental sensors and performs calculations with them every specified amount of time. These calculations and measurements are sent via satellite to a web application.</p> <p>When certain parameters are met, sensors are triggered faster to capture more data.</p>



## Hardware - Level 1

[JCM, MS, SW, NS]

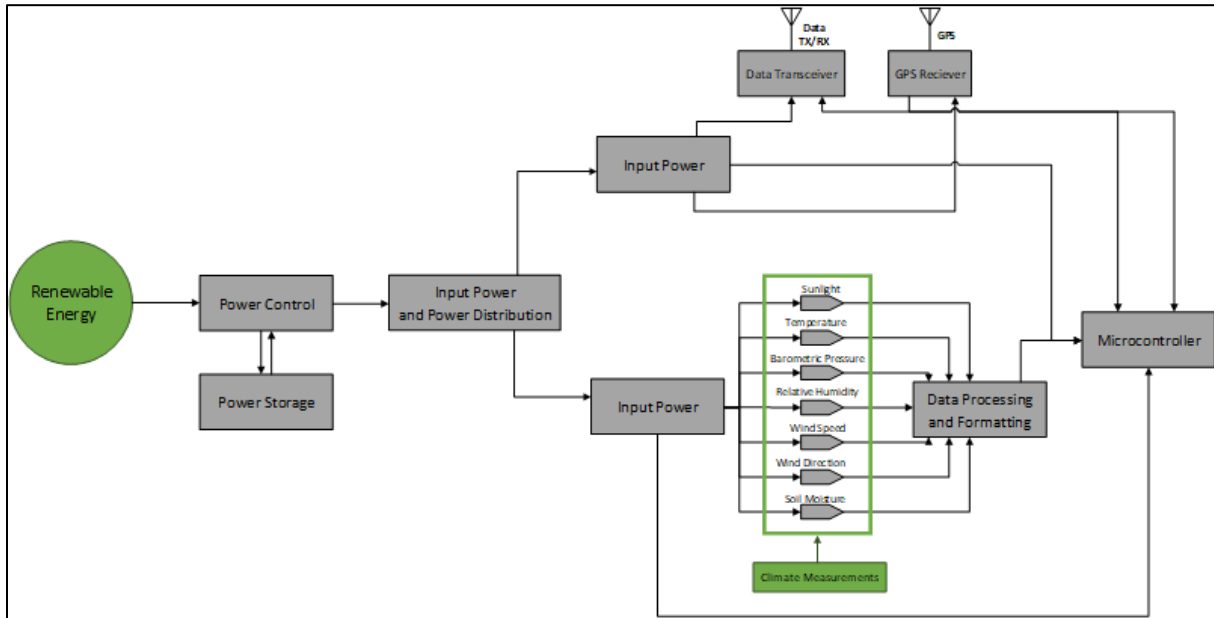


Figure 17: Level 1 Hardware Block Diagram

Table 7: Functional Requirement Table for the Level 1 Hardware Block Diagram

Module	Weather Monitor Station – Hardware Array
Designers	DPS Team 11 (JCM, NS, MS, SW)
Inputs	<ul style="list-style-type: none"> <li>- Input power (renewable energy, battery power, all controlled with regulation and filtration)</li> <li>- Environmental data (temperature, relative humidity, wind, air pressure, soil moisture)</li> <li>- GPS tracking (accurate to ~10ft)</li> <li>- Satellite data (pings, relative strength of signal, etc.)</li> </ul>

Outputs	- Satellite communication (sending of environmental data)
Functionality	<p>Takes in environmental sensors and performs calculations with them every specified amount of time. These calculations and measurements are sent via satellite to a web application.</p> <p>When certain parameters are met, sensors are triggered faster to capture more data.</p>

## Level 2 Sensor Block Diagram

[MS]

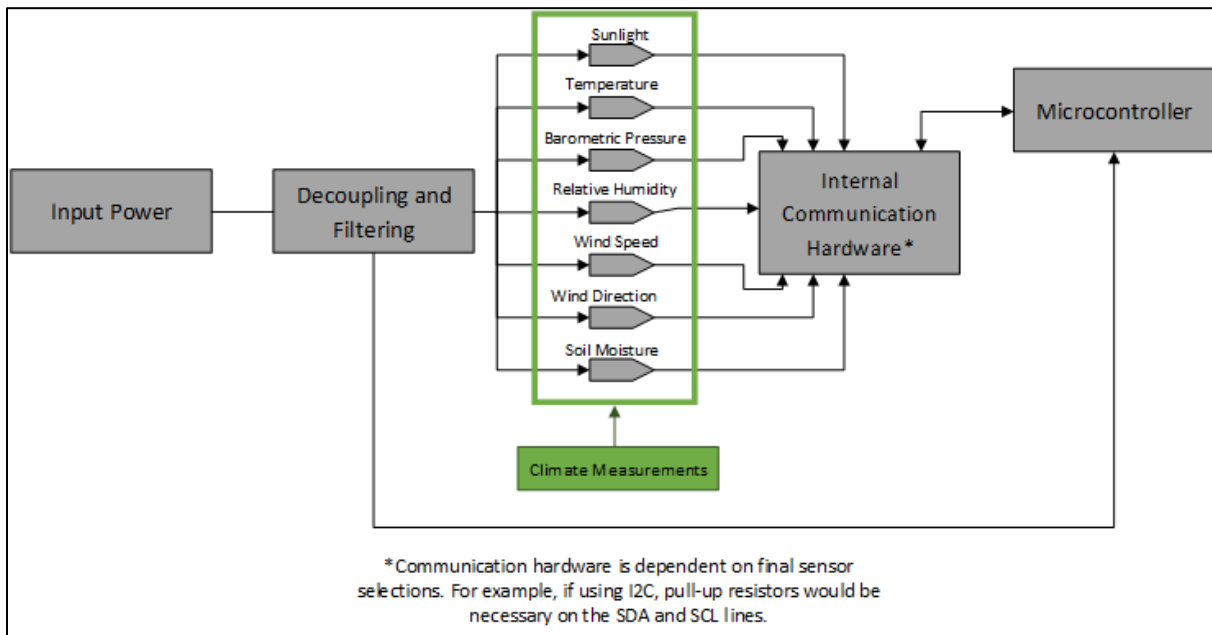


Figure 18: Level 2 Sensor Block Diagram

Table 8: Functional Requirement Table for the Level 2 Sensor Block Diagram

Module	Weather Monitor Station – Sensor Array
Designers	DPS Team 11 (JCM, NS, MS, SW)
Inputs	<ul style="list-style-type: none"> <li>- Input power (renewable energy, battery power, all controlled with regulation and filtration)</li> <li>- Environmental data (temperature, relative humidity, wind, air pressure, soil moisture)</li> <li>- GPS tracking (accurate to ~10ft)</li> </ul>
Outputs	<ul style="list-style-type: none"> <li>- Satellite communication (sending of environmental data)</li> </ul>

Functionality	<p>Takes in environmental sensors and performs calculations with them every specified amount of time. These calculations and measurements are sent via satellite to a web application.</p> <p>When certain parameters are met, sensors are triggered faster to capture more data.</p>
---------------	---

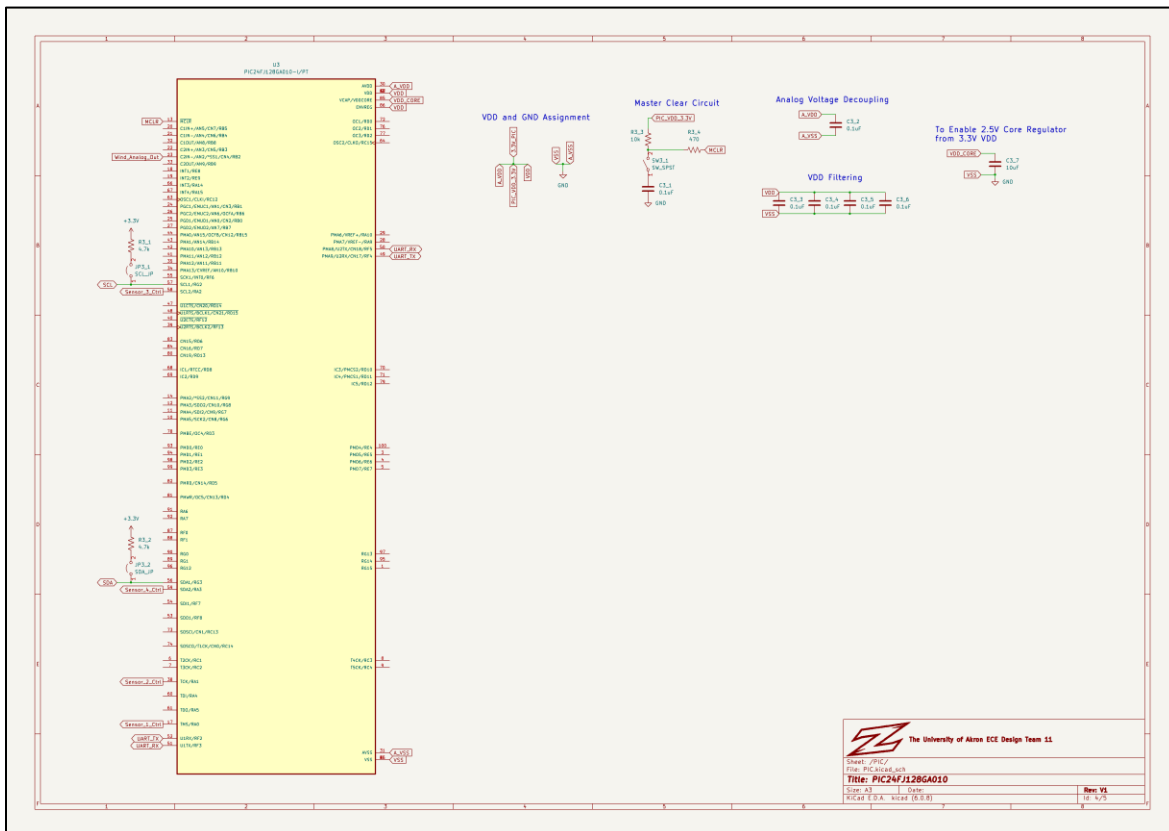


Figure 19: PIC24FJ128GA010 Microcontroller Schematic

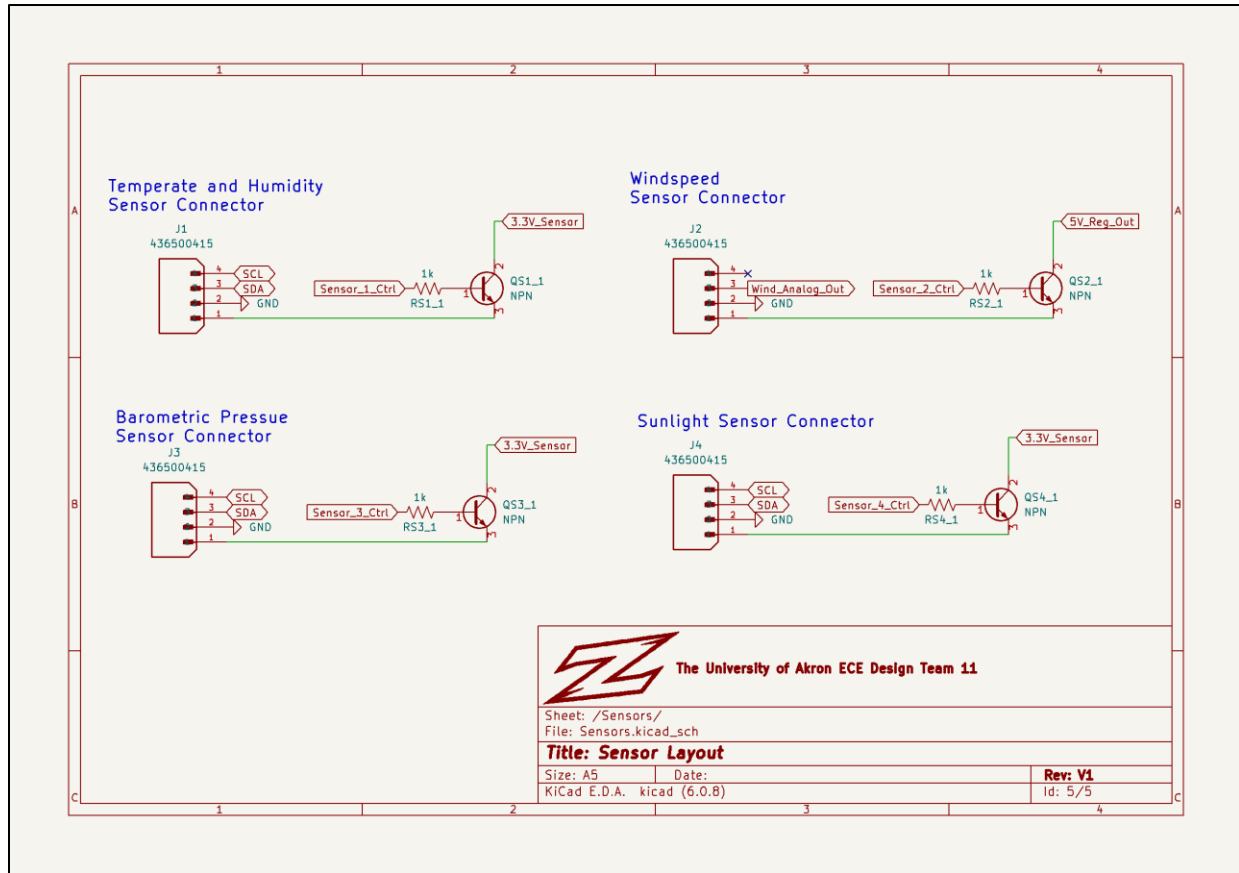


Figure 20: Sensors Control and Connection Circuit

For testing and evaluation of the various sensors, the Explorer 16/32 Development Board from Microchip was used for this task. The module that is installed on this development board contains a PIC24FJ128GA010 microcontroller which has the required I/O, communication peripherals, and analog to digital converter needed to read in data from the sensors. The barometer that was selected was the LPS33HW and utilizes I2C for communication. The temperature and humidity sensor was an SHT31 that also utilized I2C, the windspeed sensor was a generic anemometer that outputted analog voltage from 0.4V to 2.0V, and lastly the sunlight sensor is a VEML7700 Lux Sensor that implements I2C. The various development boards had built in pull-up resistors on the I2C SDA/SCL lines, so those are not implemented but jumpers are placed on the schematic incase they are needed for the final design.

## Level 2 Communication Block Diagram

[NS, MS]

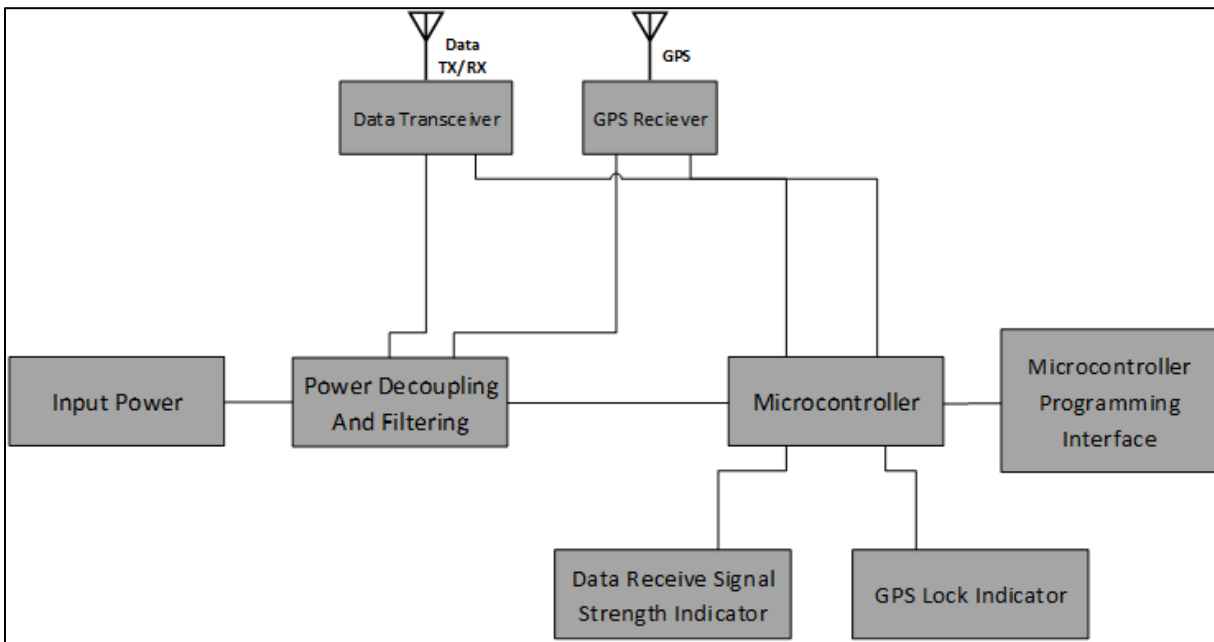


Figure 21: Level 2 Communication Block Diagram

Table 9: Functional Requirement Table for the Level 2 Communication Block Diagram

Module	Weather Monitor Station – Communication Module
Designers	DPS Team 11 (JCM, NS, MS, SW)
Inputs	<ul style="list-style-type: none"> <li>- Input power (renewable energy, battery power, all controlled with regulation and filtration)</li> <li>- Environmental data (temperature, relative humidity, wind, air pressure, soil moisture)</li> <li>- GPS tracking (accurate to ~10ft)</li> <li>- Satellite data (pings, relative strength of signal, etc.)</li> </ul>

Outputs	- Satellite communication (sending of environmental data)
Functionality	<p>Takes in environmental sensors and performs calculations with them every specified amount of time. These calculations and measurements are sent via satellite to a web application.</p> <p>When certain parameters are met, sensors are triggered faster to capture more data.</p>

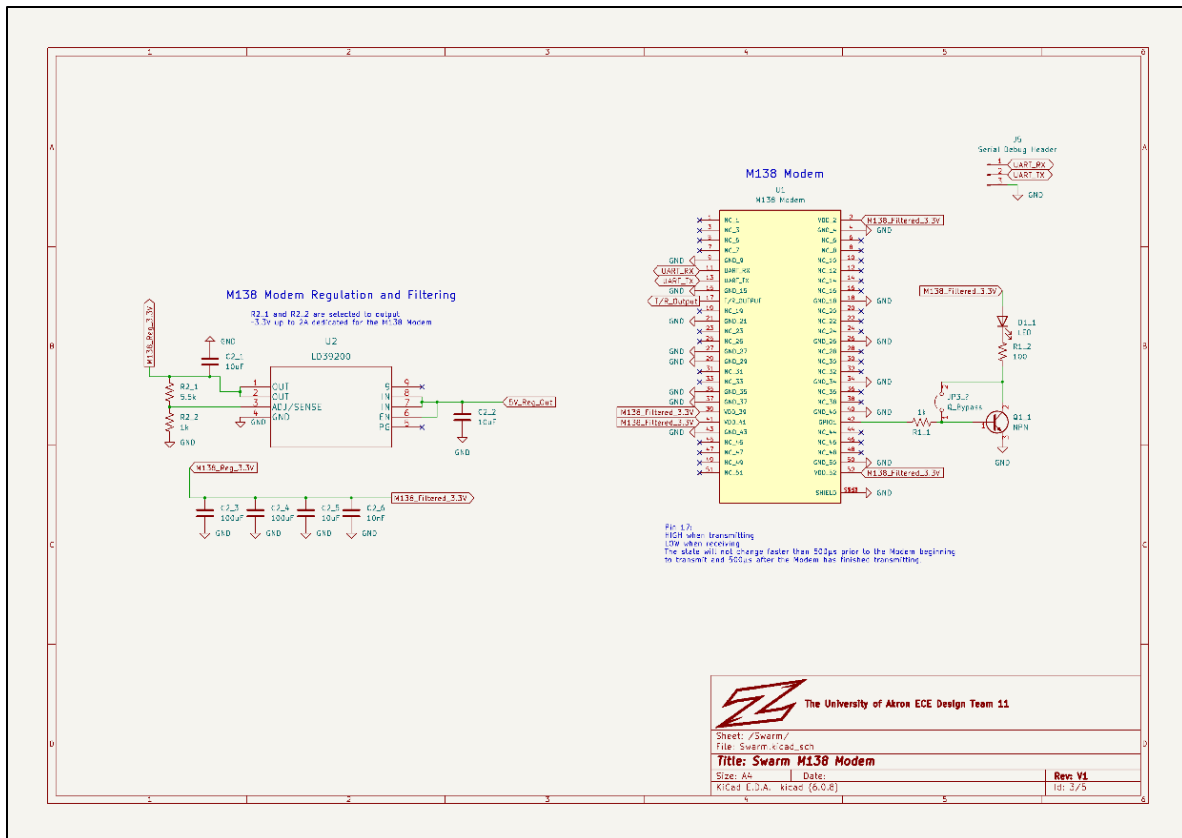


Figure 22: Swarm Modem Circuit Layout

The wireless communication sub-system consists of two main components that are distinct from the microcontroller and other voltage regulation. These being the M138 Modem

(U1) and the LD39200 (U2) adjustable voltage regulator. In combination with various supporting passive components. The LD39200 adjustable voltage regulator is a 2A regulator that has R2\_1 and R2\_2 set so that the output voltage is 3.25V. This was determined using the equation provided by ST in the LD39200 datasheet:

$$V_{OUT} = V_{ADJ} \left( 1 + \frac{R_{21}}{R_{22}} \right) \text{ or } V_{OUT} = 0.5V \left( 1 + \frac{5500\Omega}{1000\Omega} \right)$$

The input of this voltage regulator comes from the 3A 5V regulator that feeds the entire system. A separate regulator was used for this system so that it would have uninterrupted clean ~3.3V and not be affected by any other sensors that might cause current spikes which would lead to a brown out. The various filtering capacitors are used to further clean the voltage coming off the regulator and going into the M138 modem. The M138 modem utilizes two pins for UART TX/RX, a power pin, a ground pin, and then a GPIO pin that is setup to output high (3.3V) when there is a message in the queue waiting to be sent to a satellite, and then off (0V) when there are no messages in the queue. A NPN transistor was used to switch an LED on using the GPIO pin as a signal as the pin can only source roughly 8 mA of current which is not enough to drive an LED brightly. This was used for demonstration purposes and may be disabled in the final design to further conserve power. The UART TX of the M138 modem is connected to the RX of the PIC24, and the UART RX is connected to the PIC24 TX.



## Level 2 Power Block Diagram

[JCM]

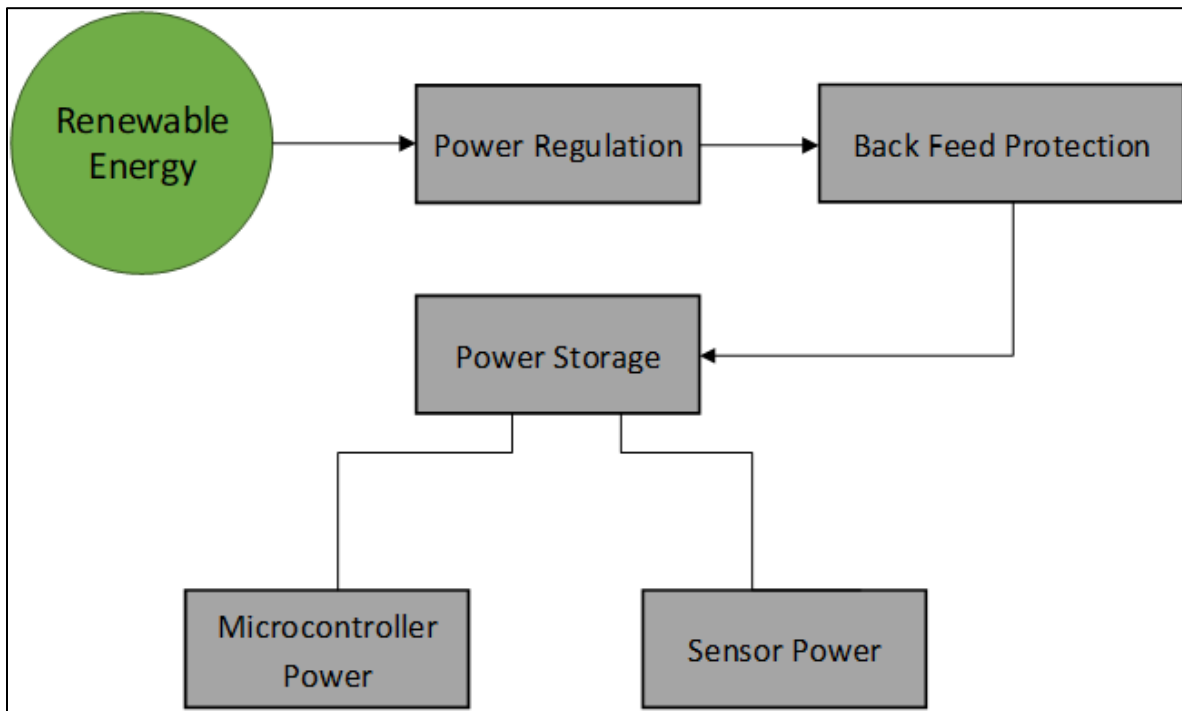


Figure 23: Level 2 Power Block Diagram

Table 10: Functional Requirement Table for the Level 2 Power Block Diagram

Module	Weather Monitor Station – Power Distribution
Designers	DPS Team 11 (JCM, NS, MS, SW)
Inputs	- Renewable energy
Outputs	- Microcontroller power - Sensor power
Functionality	Take in renewable energy, regulate, and direct it for the purposed of being fed into a power storage device, then power

	the microcontroller and the sensors from this power storage device.
--	---

### Level 3 Power Block Diagram

[JCM]

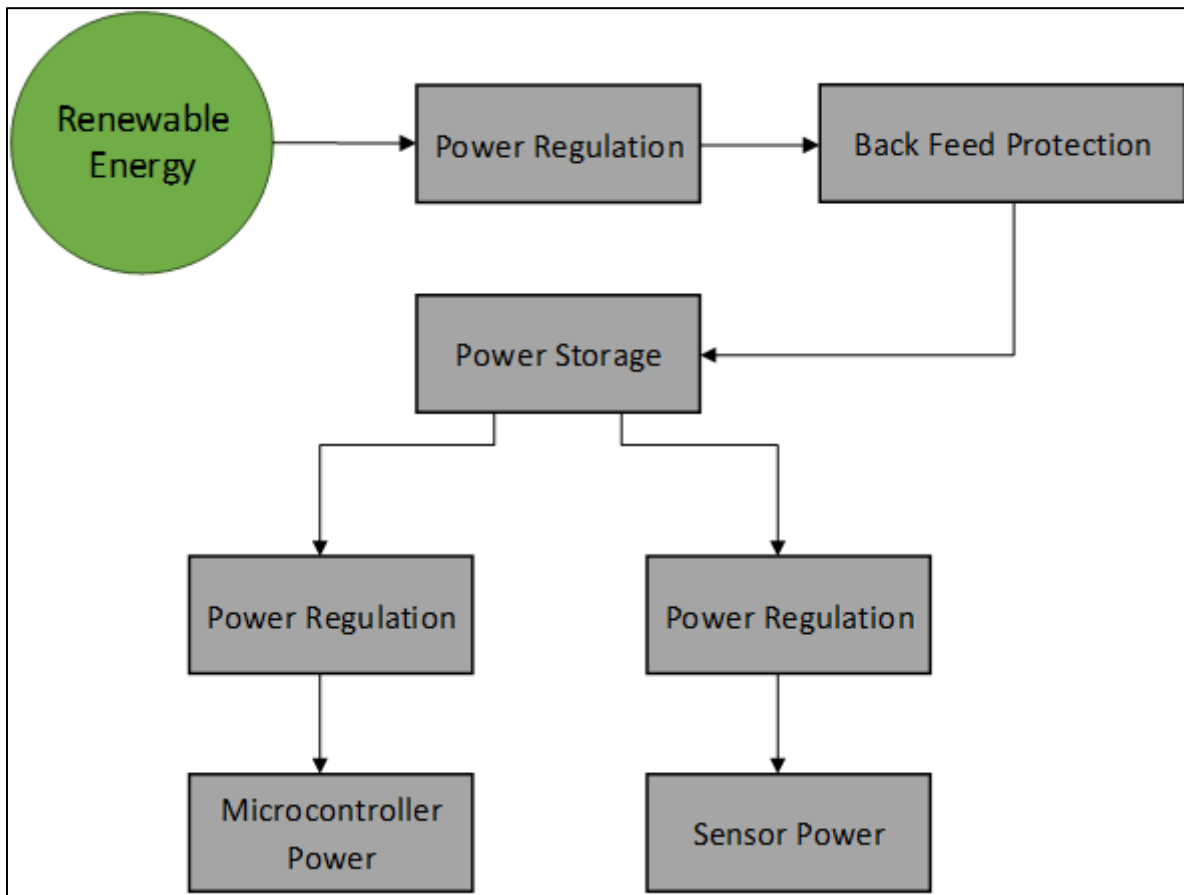


Figure 24: Level 3 Power Block Diagram

Table 11: Functional Requirement Table for the Level 3 Power Block Diagram

Module	Weather Monitor Station – Power Distribution
Designers	DPS Team 11 (JCM, NS, MS, SW)
Inputs	- Renewable energy
Outputs	- Microcontroller power - Sensor power

Functionality	Take in renewable energy, regulate, and direct it for the purposed of being fed into a power storage device. Regulate this to different levels to account for the possibility that the microcontroller and the sensors may need to be powered differently, then deliver power to these devices.
---------------	---

## Level 4 Power Block Diagram

[JCM]

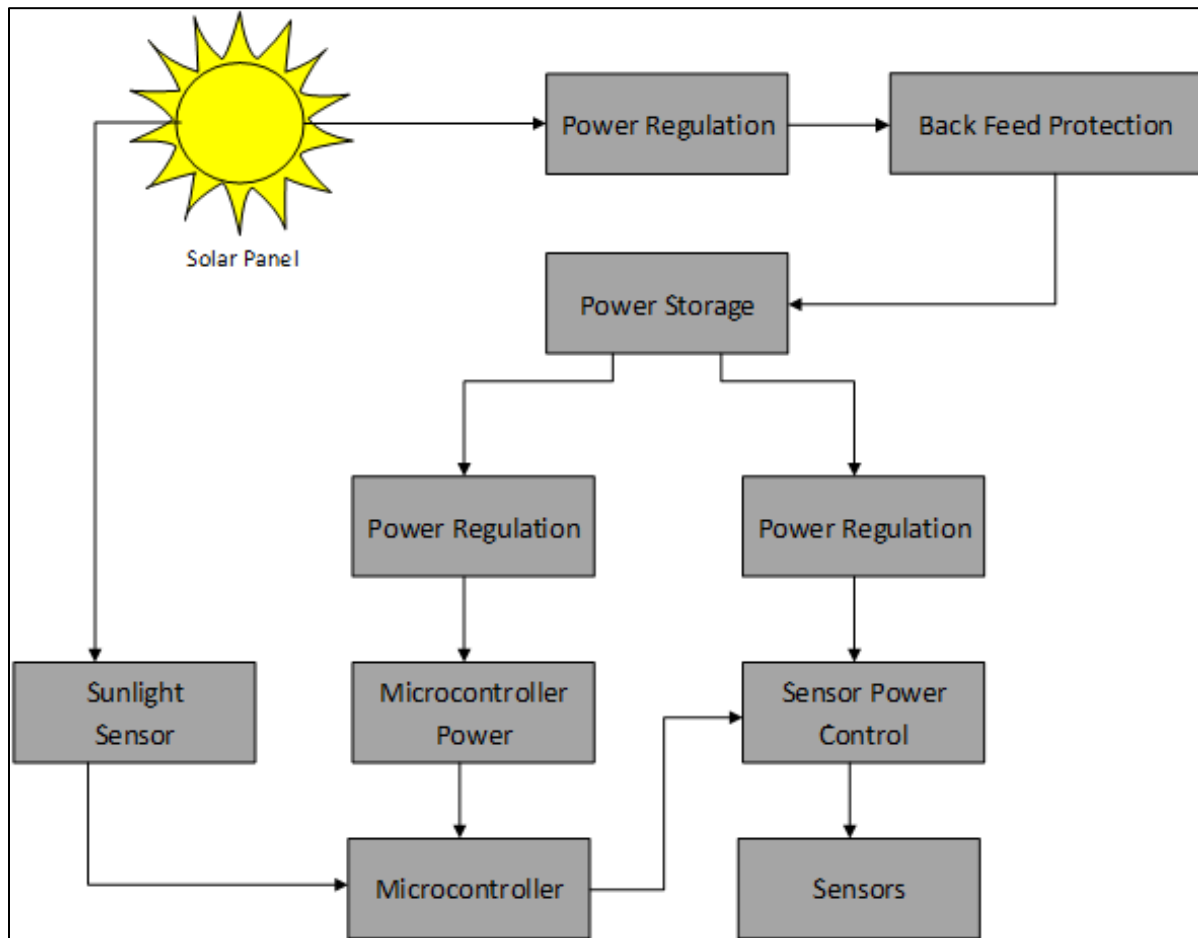


Figure 25: Level 4 Power Block Diagram

Table 12: Functional Requirement Table for the Level 4 Power Block Diagram

Module	Weather Monitor Station – Power Distribution
Designers	DPS Team 11 (JCM, NS, MS, SW)
Inputs	- Solar energy
Outputs	- Microcontroller power - Sensor power

Functionality	<p>Take in solar energy, regulate, and direct it for the purpose of being fed into a power storage device. Regulate this to different levels to account for the possibility that the microcontroller and the sensors may need to be powered differently, then deliver power to these devices. Allow the microcontroller to direct the power supplies to the sensors.</p>
---------------	--

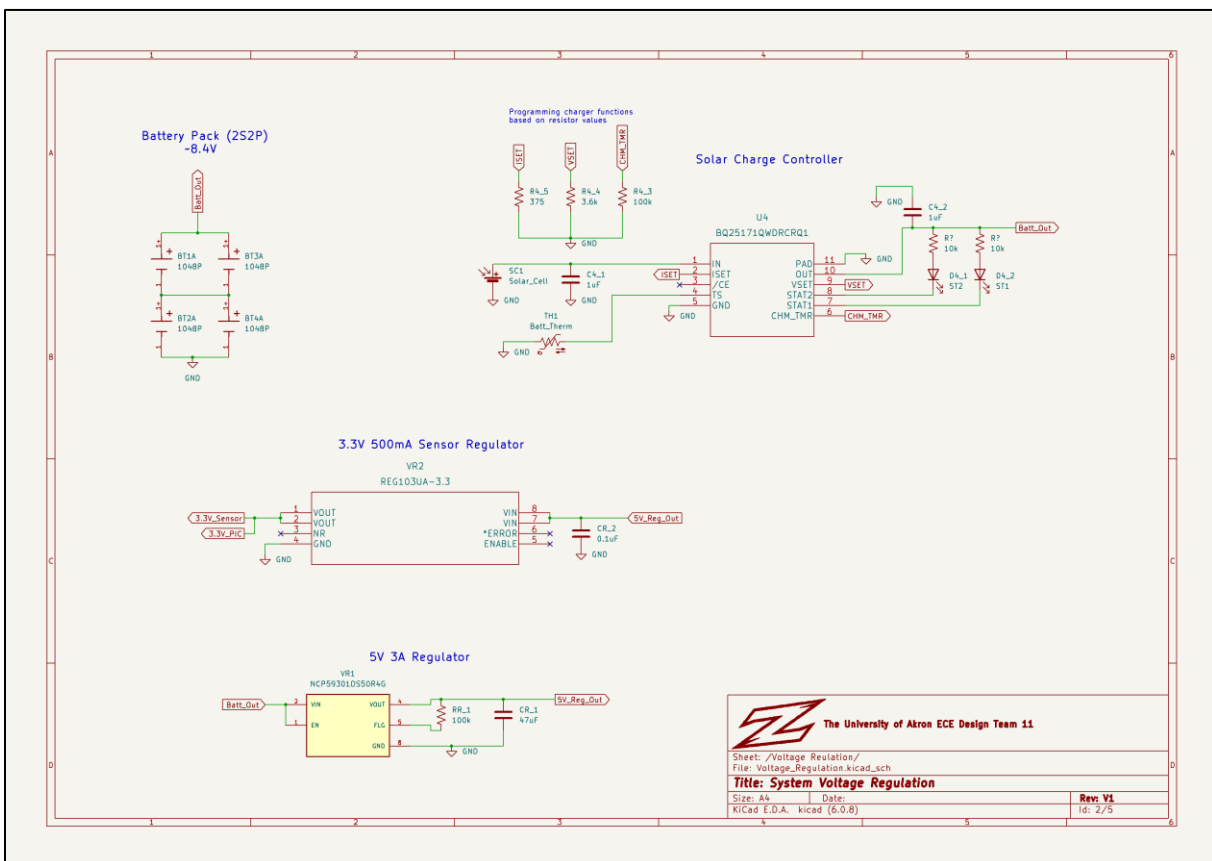


Figure 26: Power and Voltage Regulation Sub-System Schematic

The three resistors designated R4\_5, R4\_4, and R4\_3 is used to program the battery charger IC per the manufacturers data sheet. The resistor values that were used correspond to programming the IC to charge a two-cell lithium-ion battery, which is what will be used in the final design in the form of 18650 battery cells. The current resistor sets the IC to charge the batteries at a rate of 800 mA, the voltage resistor sets the output voltage to 8.4V, and then the timer resistor is selected to match the battery chemistry and charge profile to match lithium-ion cells. The battery charger will receive raw unregulated voltage from the solar panel and then output a steady 8.4V to the batteries. Two status LEDs are used to indicate the status of charging and if there are any errors present when charging. A NTC thermistor is also required for this charging IC and this is used to monitor the temperate of the cells for safety. From the output of the charging IC and battery, the voltage is then regulated down through a 3A 5V regulator which will power the entire system. The M138 modem requires 2A of dedicated power, so the 1A that is left over will power the sensors and microcontroller. All sensors except for the windspeed sensor utilize 3.3V, thus the need for a separate 3.3V 500mA regulator. The microcontroller then controls the flow of power into each sensor using a transistor as shown in the sensor subsystem section, this way further energy consumption can be limited when the sensors are not being polled for environmental data. Further considerations for how much power is expected to be consumed by the system can be found in the Power Distribution section.

## 5.2 Software Design

### Level 0 Software Block Diagram

[SW, NS]

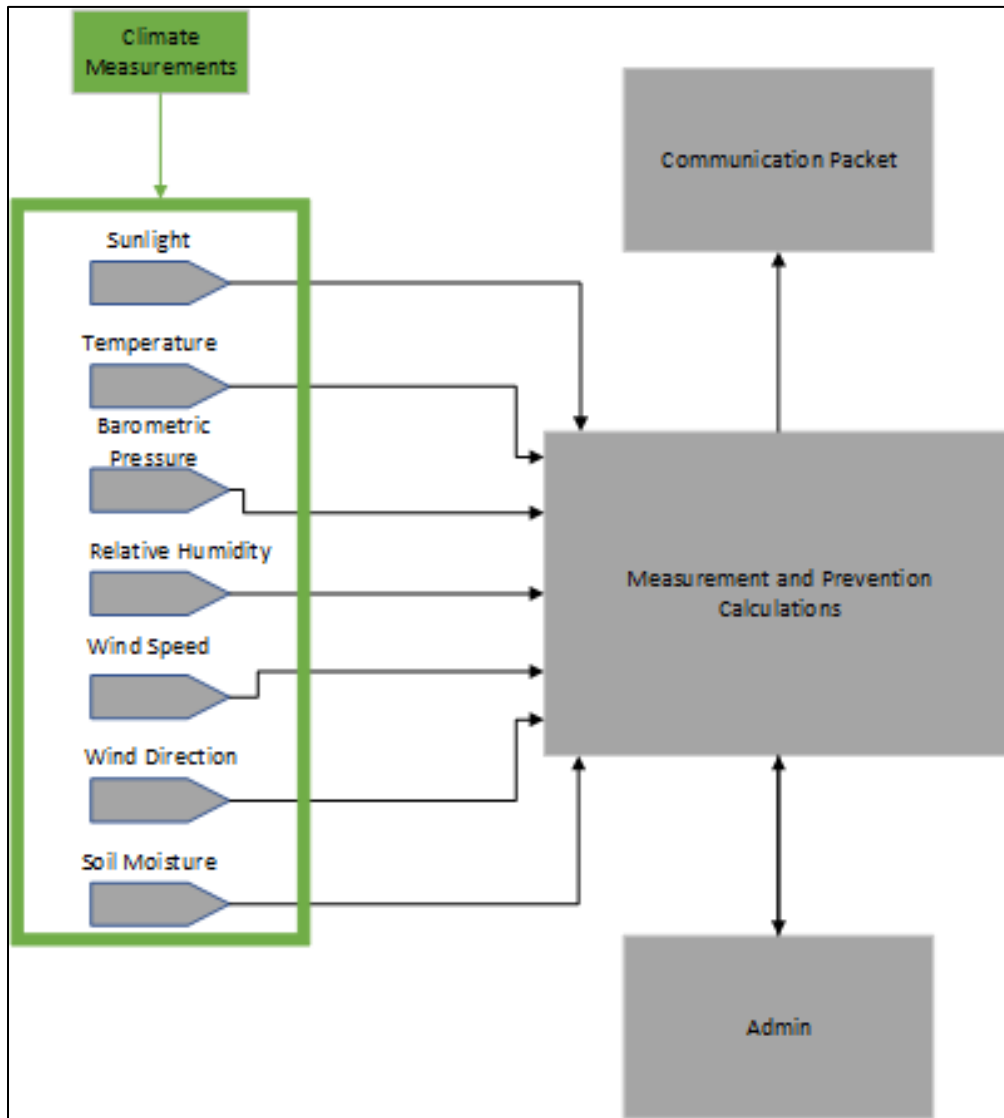


Figure 27: Software Level 0 Block Diagram



Table 13: Functional Requirement Table for the Level 0 Software Block Diagram

Module	Weather Monitor Station - Program (level 0)
Designers	DPS Team 11 (JCM, NS, MS, SW)
Inputs	<ul style="list-style-type: none"> <li>- Temperature data</li> <li>- Wind data (speed, direction)</li> <li>- Humidity data</li> <li>- Soil Moisture data</li> <li>- Administrative inputs</li> </ul>
Outputs	- Packet containing data of recorded measurements and result of Wildfire occurrence plausibility calculation
Functionality	Take in environmental data to communicate the current weather conditions to an end user.

## Level 1 Software Block Diagram

[SW, NS]

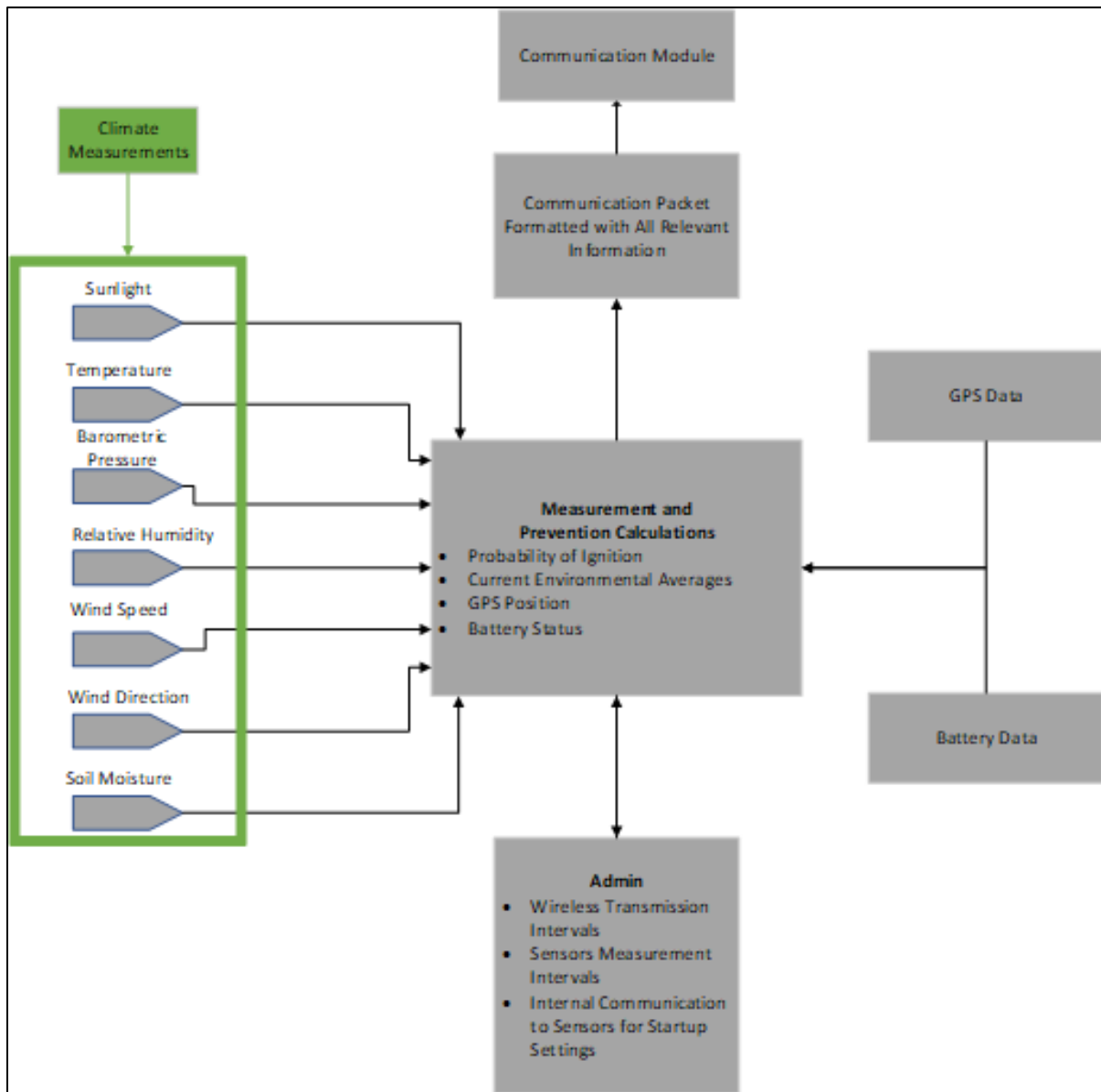


Figure 28: Software Level 1 Block Diagram

Table 14: Functional Requirement Table for the Level 1 Software Block Diagram

Module	Weather Monitor Station - Program (level 1)
Designers	DPS Team 11 (JCM, NS, MS, SW)
Inputs	<ul style="list-style-type: none"> <li>- Temperature data</li> <li>- Wind data (speed, direction)</li> <li>- Humidity data</li> <li>- Soil Moisture data</li> <li>- Battery data</li> <li>- GPS data</li> <li>-Administrative inputs</li> </ul>
Outputs	<ul style="list-style-type: none"> <li>- Packet containing data of recorded measurements and result of Wildfire occurrence plausibility calculation.</li> </ul>
Functionality	Take in environmental data, GPS data, and battery data to communicate the current weather conditions, location, and battery status of the weather monitor to an end user.

## Detailed Software Flowchart

[SW, NS]

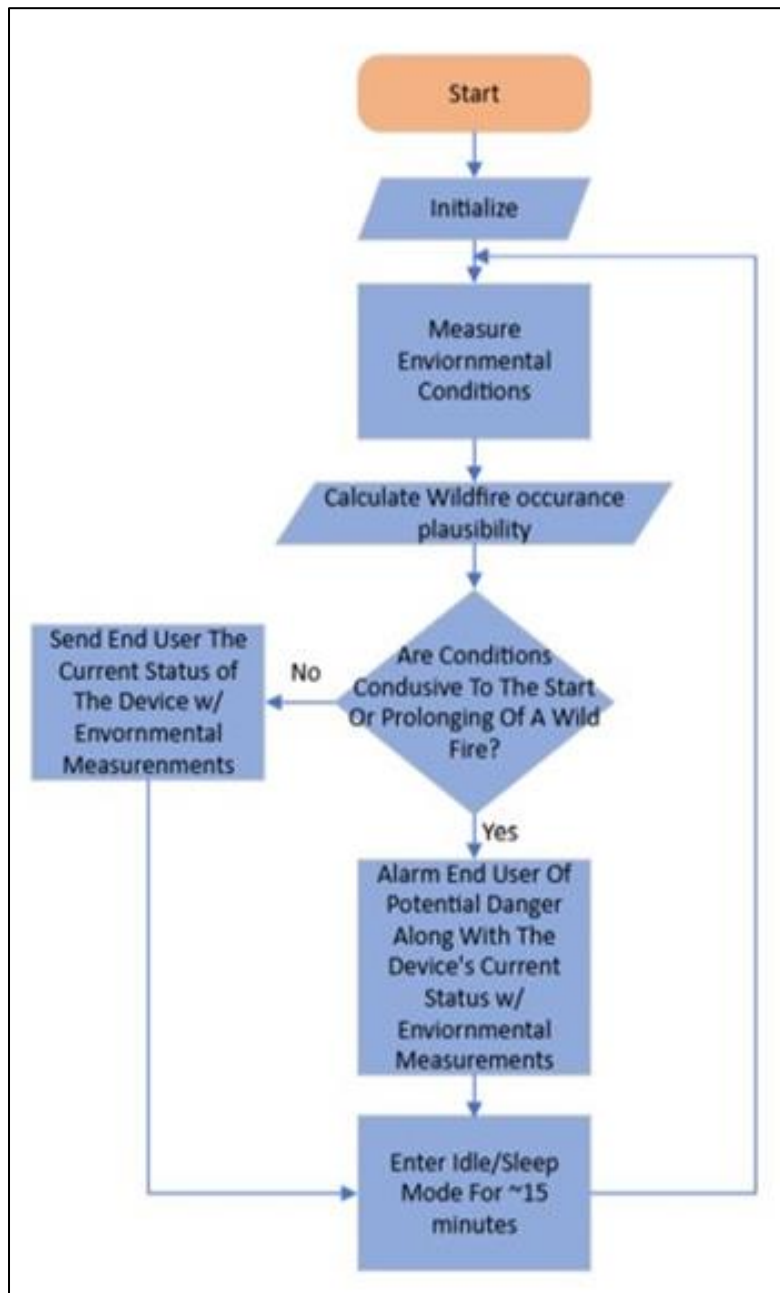


Figure 29: Software High Level Flowchart

## 6. Mechanical Design

[NS]

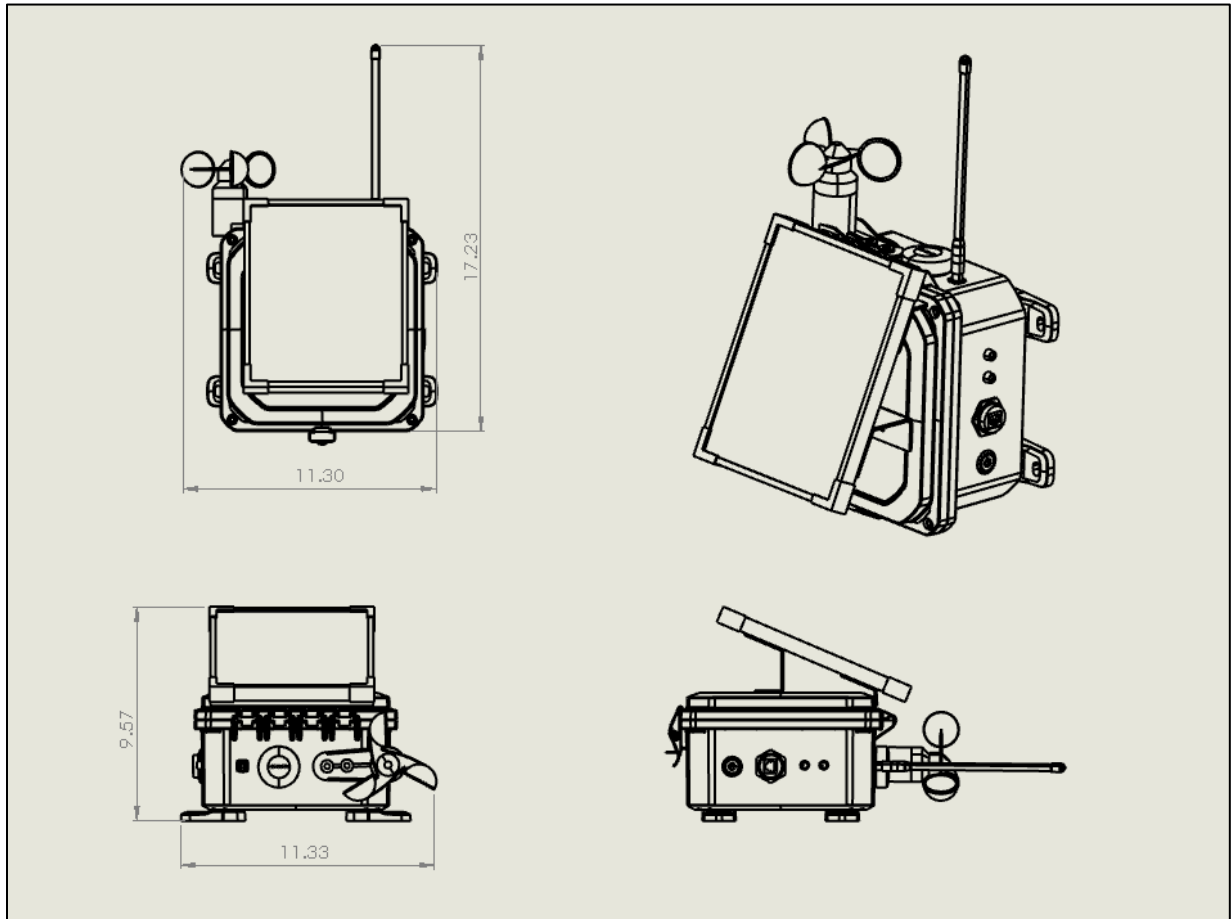


Figure 30: Mechanical Drawing with Estimated Dimensions.<sup>1</sup>

<sup>1</sup> All dimension in inches

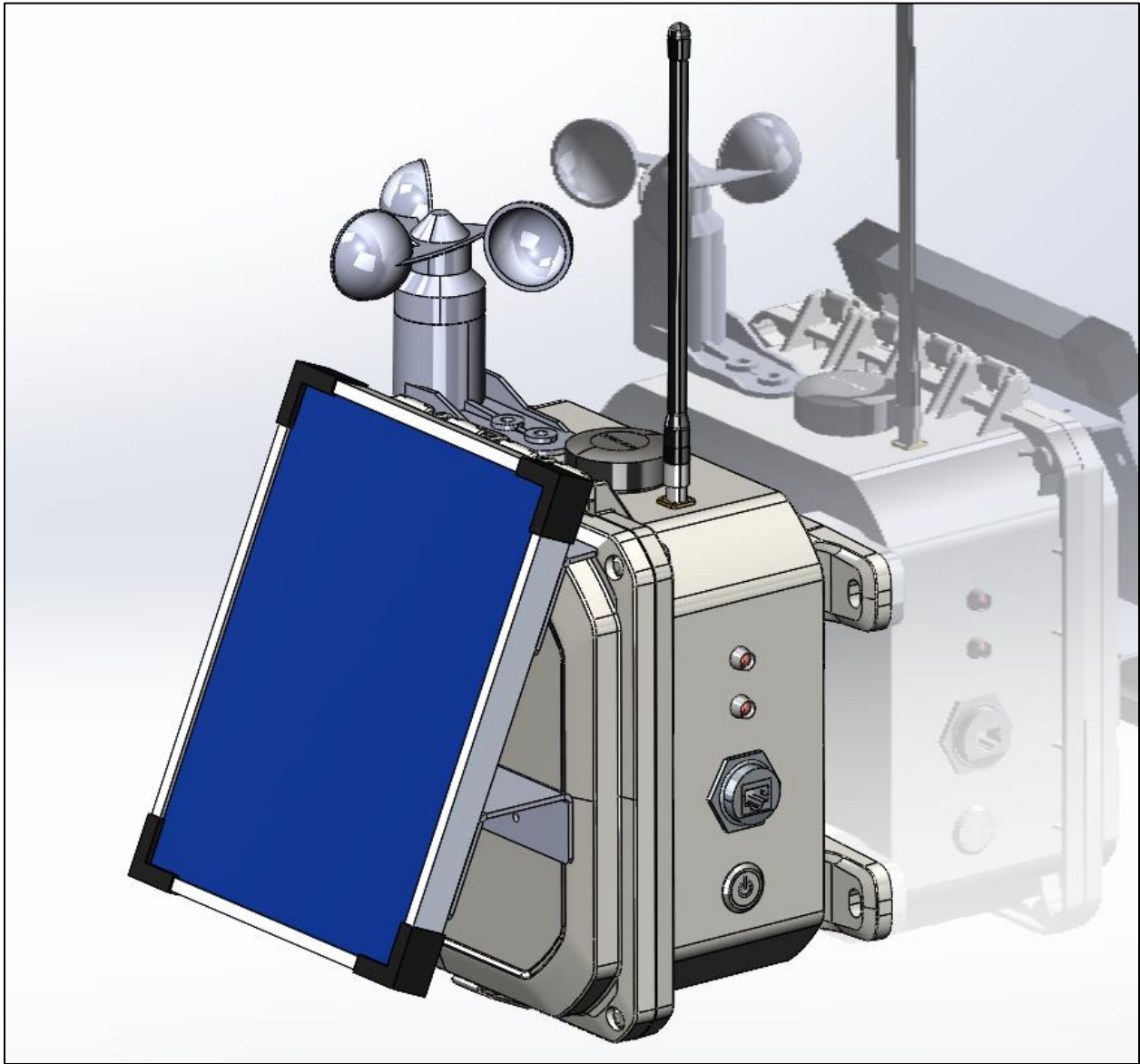


Figure 31: Mechanical 3D Rendering

## 7. Team Information

Matthew Szijarto: Electrical Engineering

Joel Christie-Millett: Electrical Engineering

Sylvester Wilson: Computer Engineering

Nathan Schroeder: Computer Engineering

## 8. Parts List

### 8.1. Parts List for Accepted Technical Design

[NS]

Table 15: Parts List

Item	Qty	Reference(s)	Value	MANUFACTURER
1	4	BT1, BT2, BT3, BT4	1048P	Keystone
2	4	C2_1, C2_2, C2_5, C3_7	10uF	Any
3	2	C2_3, C2_4	100uF	Any
4	1	C2_6	10nF	Any
5	7	C3_1, C3_2, C3_3, C3_4, C3_5, C3_6, CR_2	0.1uF	Any
6	2	C4_1, C4_2	1uF	Any
7	1	CR_1	47uF	Any
8	1	D1_1	LED	Any
9	1	D4_1	ST2	Any
10	1	D4_2	ST1	Any
11	4	J1, J2, J3, J4	436500415	Molex
12	1	J5	3x 2.54mm Header	Any
13	1	JP3_1	SCL_JP	Any
14	1	JP3_2	SDA_JP	Any
15	5	Q1_1, QS1_1, QS2_1, QS3_1, QS4_1	NPN	Any
16	3	R1, R2, R3_3	10k	Any
17	6	R1_1, R2_2, RS1_1, RS2_1, RS3_1, RS4_1	1k	Any
18	1	R1_2	100	Any
19	1	R2_1	5.5k	Any

20	2	R3_1, R3_2	4.7k	Any
21	1	R3_4	470	Any
22	2	R4_3, RR_1	100k	Any
23	1	R4_4	3.6k	Any
24	1	R4_5	375	Any
25	1	SC1	Solar Cell	Any
26	1	SW3_1	SW_SPST	Any
27	1	TH1	Battery Thermistor	Any
28	1	U1	M138 Modem	Swarm.Space
29	1	U2	LD39200	ST
30	1	U3	PIC24FJ128GA010-I/PT	Microchip
31	1	U4	BQ25171QWDRCRQ1	Texas Instruments
32	1	VR1	NCP59301DS50R4G	ON Semiconductor
33	1	VR2	REG103UA-3.3	Texas Instruments

## 8.2. Materials Budget

[NS]

Table 16: Materials Budget

Qty.	Part Num.	Description	Individual Cost	Total Cost
1	BQ25171QWDRCRQ1	Battery Management Automotive, 800-mA linear battery charger for 1- to 2-cell Li-ion, LiFePO4	\$2.13	\$2.13
1	5378	VEML7700 Light Sensor Breakout Board	\$4.95	\$4.95
1	BOB-00544	MicroSD Card Breakout Board	\$5.50	\$5.50
1	IPC0067	DFN-10 TO DIP-14 SMT ADAPTER	\$5.79	\$5.79
1	PA0185	TO-263-5 To DIP-10 Adapter	\$4.79	\$4.79
5	TCA9517DR	3.3V to 5V bidirectional Level Shifter	\$1.28	\$6.40
5	LCQT-SOIC8-8	SOIC to DIP Adapter	\$3.36	\$16.80
2	IPC0065	DFN-8 (4mmx4mm) To DIP Adapter	\$5.29	\$10.58



2	LD39200DPUR	Adjustable 2A Linear Voltage Regulator	\$1.27	\$2.54
1	IPC0061	DFN-8 to DIP-12 SMT Adapter	\$5.29	\$5.29
1	4414	LPS33HW Barometric Pressure Breakout Board	\$12.50	\$12.50
1	TPS7333QP	500mA 3.3V Linear Voltage Regulator	\$3.58	\$3.58
1	REG103UA-3.3	500mA 3.3V Linear Voltage Regulator	\$6.53	\$6.53
1	ATGG46015-BP-SMA-3	GPS Antenna Puck	\$12.64	\$12.64
1	SEN0385	SHT31 I2C Temperature and Humidity Sensor	\$19.90	\$19.90
2	LCQT-SOIC8-8	8-SOIC to 8 DIP SMT Adapter	\$3.36	\$6.72
1	313070005	1W 8.2V Solar Cell	\$12.30	\$12.30
2	BQ2057WSN	8.4V Battery Charger IC	\$4.82	\$9.64
1	L74A52-4-10-2WX	7.4 VOLT Battery	\$26.44	\$26.44
<b>Total</b>				<b>\$175.02</b>

## 9. Project Schedules

SDP I 2022				
Parts Request Form for Spring Semester	0 days	Fri 12/2/22	Fri 12/2/22	
▣ <b>Project Design</b>	<b>95 days?</b>	<b>Wed 8/24/22</b>	<b>Sun 11/27/22</b>	
<b>Final Design Report</b>	47 days	Tue 10/11/22	Sun 11/27/22	Whole Team
<b>Abstract</b>	47 days	Tue 10/11/22	Sun 11/27/22	Whole Team
<b>Project Poster</b>	14 days	Tue 10/11/22	Tue 10/25/22	Whole Team
<b>Midterm Parts Request Form</b>	50 days	Wed 8/24/22	Thu 10/13/22	Whole Team
<b>Midterm Report</b>	46 days	Wed 8/24/22	Sun 10/9/22	Whole Team
Cover page	46 days	Wed 8/24/22	Sun 10/9/22	Whole Team
T of C, L of T, L of F	46 days	Wed 8/24/22	Sun 10/9/22	Whole Team
▣ <b>Problem Statement</b>	<b>46 days</b>	<b>Wed 8/24/22</b>	<b>Sun 10/9/22</b>	
Need	46 days	Wed 8/24/22	Sun 10/9/22	Whole Team
Objective	46 days	Wed 8/24/22	Sun 10/9/22	Whole Team
Background	46 days	Wed 8/24/22	Sun 10/9/22	Whole Team
Marketing Requirements	46 days	Wed 8/24/22	Sun 10/9/22	Whole Team
Engineering Requirements Specification	46 days	Wed 8/24/22	Sun 10/9/22	Whole Team
▣ <b>Engineering Analysis</b>	<b>46 days?</b>	<b>Wed 8/24/22</b>	<b>Sun 10/9/22</b>	
▣ <b>Circuits (DC, AC, Power, ...)</b>	<b>20.38 days</b>	<b>Wed 8/24/22</b>	<b>Tue 9/13/22</b>	
Sensor Power Research	5 days	Wed 8/24/22	Tue 9/13/22	Matt
Power Storage	5 days	Wed 8/24/22	Tue 9/13/22	Joel
Power Supply for Sensors, Microcontroller, Communication system, etc.	5 days	Wed 8/24/22	Tue 9/13/22	Joel
Renewable Power Research	3 days	Wed 8/24/22	Mon 9/5/22	Joel
Voltage Level Selections	3 days	Wed 8/24/22	Mon 9/5/22	Joel, Matt
Microcontroller Implementation Research	3 days	Wed 8/24/22	Mon 9/5/22	Sylvester
▣ <b>Electronics (analog and digital)</b>	<b>20.38 days</b>	<b>Wed 8/24/22</b>	<b>Tue 9/13/22</b>	
Microcontroller Research	5 days	Wed 8/24/22	Tue 9/13/22	Sylvester
Power Supply and Filtering for Microcontroller	5 days	Wed 8/24/22	Tue 9/13/22	Joel

Figure 32: Semester 1 Project Schedule

Sensor Research	5 days	Wed 8/24/22	Tue 9/13/22	Matt
▣ <b>Communications (analog and digital)</b>	<b>20.38 days</b>	<b>Wed 8/24/22</b>	<b>Tue 9/13/22</b>	
Primary Communication System Research and Feasibilit	5 days	Wed 8/24/22	Tue 9/13/22	Nate
Antenna Research	3 days	Wed 8/24/22	Mon 9/5/22	Nate
GPS Antenna Research	3 days	Wed 8/24/22	Mon 9/5/22	Nate
▣ <b>Embedded Systems</b>	<b>20.38 days</b>	<b>Wed 8/24/22</b>	<b>Tue 9/13/22</b>	
Research Various Communication Modules (Pros and Co	5 days	Wed 8/24/22	Tue 9/13/22	Nate
▣ <b>Signal Processing and Formatting</b>	<b>15.38 days</b>	<b>Wed 8/24/22</b>	<b>Thu 9/8/22</b>	
Research Wired Communication Protocols for Sensor In	4 days	Wed 8/24/22	Thu 9/8/22	Matt,Nate,Sylvester
▣ <b>Electromechanics</b>	<b>12.38 days</b>	<b>Wed 8/24/22</b>	<b>Mon 9/5/22</b>	
Rugged Enclosure Research	3 days	Wed 8/24/22	Mon 9/5/22	Nate
▣ <b>Accepted Technical Design</b>	<b>46 days?</b>	<b>Wed 8/24/22</b>	<b>Sun 10/9/22</b>	
▣ <b>Hardware Design: Phase 1</b>	<b>46 days?</b>	<b>Wed 8/24/22</b>	<b>Sun 10/9/22</b>	
▣ <b>Hardware Block Diagrams Levels 0 thru N (w/ FR tables)</b>	<b>5 days</b>	<b>Wed 8/24/22</b>	<b>Mon 8/29/22</b>	<b>Whole Team</b>
Communication Block Diagram	5 days	Wed 8/24/22	Mon 8/29/22	
Sensor Block Diagram	5 days	Wed 8/24/22	Mon 8/29/22	
Power Block Diagram	5 days	Wed 8/24/22	Mon 8/29/22	
▣ <b>Software Design: Phase 1</b>	<b>46 days</b>	<b>Wed 8/24/22</b>	<b>Sun 10/9/22</b>	
Software Behavior Models Levels 0 thru N (w/FR tables)	46 days	Wed 8/24/22	Sun 10/9/22	Whole Team
<b>Mechanical Sketch</b>	46 days	Wed 8/24/22	Sun 10/9/22	Whole Team
<b>Team information</b>	46 days	Wed 8/24/22	Sun 10/9/22	Whole Team
▣ <b>Project Schedules</b>	<b>46 days</b>	<b>Wed 8/24/22</b>	<b>Sun 10/9/22</b>	
Midterm Design Gantt Chart	19 days	Wed 8/24/22	Mon 9/12/22	Whole Team
<b>References</b>	46 days	Wed 8/24/22	Sun 10/9/22	Whole Team
<b>Midterm Design Presentations Day 2</b>	0 days	Wed 10/5/22	Wed 10/5/22	Whole Team
<b>Midterm Design Presentations Day 1</b>	0 days	Wed 9/28/22	Wed 9/28/22	Whole Team
Midterm presentation file submission	33 days	Wed 8/24/22	Mon 9/26/22	Whole Team

Figure 33: Project Schedule cont.

▸ <b>Hardware Design: Phase 2</b>	<b>47 days</b>	<b>Tue 10/11/22</b>	<b>Sun 11/27/22</b>
▸ <b>Modules 1...n</b>	<b>47 days</b>	<b>Tue 10/11/22</b>	<b>Sun 11/27/22</b>
Simulations	47 days	Tue 10/11/22	Sun 11/27/22
Schematics	47 days	Tue 10/11/22	Sun 11/27/22
▸ <b>Software Design: Phase 2</b>	<b>47 days</b>	<b>Tue 10/11/22</b>	<b>Sun 11/27/22</b>
▸ <b>Modules 1...n</b>	<b>47 days</b>	<b>Tue 10/11/22</b>	<b>Sun 11/27/22</b>
Code (working subsystems)	47 days	Tue 10/11/22	Sun 11/27/22
System integration Behavior Models	47 days	Tue 10/11/22	Sun 11/27/22
▸ <b>Parts Lists</b>	<b>47 days</b>	<b>Tue 10/11/22</b>	<b>Sun 11/27/22</b>
Parts list(s) for Schematics	47 days	Tue 10/11/22	Sun 11/27/22
Materials Budget list	47 days	Tue 10/11/22	Sun 11/27/22
<b>Proposed Implementation Gantt Chart</b>	47 days	Tue 10/11/22	Sun 11/27/22
<b>Conclusions and Recommendations</b>	47 days	Tue 10/11/22	Sun 11/27/22
Subsystems Demonstrations Day 2	0 days	Wed 11/16/22	Wed 11/16/22
Subsystems Demonstrations Day 1	0 days	Wed 11/9/22	Wed 11/9/22
Parts Request Form for Subsystems	30.38 days	Wed 9/21/22	Fri 10/21/22

Figure 34: Project Schedule cont.

## 10. Conclusion

[NS]

To summarize, the objective of this project is to monitor the conditions of the environment for the goal of early wildfire prevention and situational awareness. The device will be able to communicate with users, using a remote application or website, to display the status of the environment and wildfire probability at the sensor's location. In turn, the application will notify users if the conditions constitute the possibility of an emergency or are favorable for wildfire conditions. Multiple of these devices can be wirelessly connected to provide users greater accuracy, range, and awareness of remote environments, as well as providing a safety net in case one sensor is destroyed. This will allow for redundancy in the case of life-threatening emergencies and allow the users the most accurate and up to date information regarding the environment around them.


## 11. References

- [1] Akyildiz, Ian Fuat, and Xudong Wang. Essay. In *Wireless Mesh Networks*, 1–9. Chichester, U.K.: Wiley, 2009.
- [2] Peterson, L. and Davie, B., n.d. *Computer networks a systems approach*. 5th ed.
- [3] Govil, K; Welch, M; Ball, J.; and Pennypacker, C. - *Preliminary Results from a Wildfire Detection System Using Deep Learning on Remote Camera Images* 2020. [Online]. Available: <https://www.mdpi.com/2072-4292/12/1/166/htm>. Accessed: 3/11/2022.
- [4] California Department of Forestry and Fire Protection. “Equipment Fact Sheet.” <https://www.fire.ca.gov/> <https://www.fire.ca.gov/media/dcjinvk/mcc-resized.pdf>. Accessed 3/11/2022.
- [5] Billing, P *Otways Fire No. 22 – 1982/83 Aspects of fire behaviour. Research Report No.20* . Victoria Department of Sustainability and Environment. 1983. [Online]. Available: [https://www.ffm.vic.gov.au/\\_data/assets/pdf\\_file/0007/21022/Report-20-Otways-Fire-22-1982-83 -Aspects-of-Fire-Behaviour.pdf](https://www.ffm.vic.gov.au/_data/assets/pdf_file/0007/21022/Report-20-Otways-Fire-22-1982-83-Aspects-of-Fire-Behaviour.pdf). Accessed: 3/11/2022.
- [6] Alberta Agriculture and Forestry. “Wildfire Detection.” <https://www.wildfire.alberta.ca/> <https://wildfire.alberta.ca/operations/wildfire-detection/default.aspx>. Accessed 3/11/2022.
- [7] Bor, Martin, et al. “Lora for the Internet of Things.” Lancaster EPrints, Junction Publishing, 15 Feb. 2016, <https://eprints.lancs.ac.uk/id/eprint/77615/>.
- [8] Nurwarsito, Heru, and Awang Satya Kusuma. “Development of Multipoint Lora Communication Network on Microclimate Datalogging System with Simple Lora Protocol.” IEEE Xplore, 29 July 2021, <https://ieeexplore.ieee.org/document/9538705/authors>.
- [9] Outdoor Wireless Communication System Based on LoRa Ad-Hoc Network. 14 July 2020. CN105813099B.

- [10] Outdoor Wireless SOS Apparatus. 15 Jan. 2014. Chinese Patent. CN203399096U.
- [11] “Climate Change Indicators: Wildfires.” *EPA*, Environmental Protection Agency,  
<https://www.epa.gov/climate-indicators/climate-change-indicators-wildfires>.
- [12] “Sun Hours Map: How Many Sun Hours Do You Get?” <https://unboundsolar.com/solar-information/sun-hours-us-map>
- [13] “Probability of Ignition” <https://www.nwcg.gov/publications/pms437/fuel-moisture/probability-of-ignition>
- [14] “Fine Dead Fuel Moisture” <https://www.nwcg.gov/publications/pms437/fuel-moisture/dead-fuel-moisture-content>
- [15] “Swarm Satellite Pass Checker” <https://kube.tools.swarm.space/pass-checker/>
- [16] “GPS Accuracy” <https://www.gps.gov/systems/gps/performance/accuracy/#how-accurate>
- [15] “Swarm Satellite Pass Checker” <https://kube.tools.swarm.space/pass-checker/>
- [17] “Illuminance - Recommended Light Level” [https://www.engineeringtoolbox.com/light-level-rooms-d\\_708.html](https://www.engineeringtoolbox.com/light-level-rooms-d_708.html)


## 12. Appendices

[NS]



PRODUCT OVERVIEW

# Swarm M138 Modem



1.2 in  
30 mm

**KEY FEATURES**

- Remote 2-way data transfer from anywhere on Earth via the Swarm constellation
- mPCIe connection provides simple integration with a PCB
- Compact, lightweight, and low-power
- Wide input voltage (3.0 V to 5.0 V)

The Swarm M138 Modem transmits and receives satellite data to and from Swarm's space network and is designed to be embedded into a third-party product. It is suitable for a variety of low-bandwidth, latency-tolerant use cases: from tracking vehicles, ships, and packages to relaying sensor data for agriculture, energy, and industrial IoT applications.

**SMALL SIZE & SIMPLE INTEGRATION**

The M138 is a miniaturized module that is designed to be embedded into any new or existing PCB design. The M138's standard mPCIe form factor makes for easy integration and replacement. It communicates via a standard serial UART or a developer-provided PC interface with a USB-to-serial converter.

**EASY TO RETRIEVE DATA**

Swarm backend systems can support delivery of customer data via a REST API or Webhook to/from the Swarm cloud or user email, text message, AWS, or Slack.

**LOW POWER**

The Swarm Modem supports a number of low-power modes which can be triggered for wake-up via built-in timer, external GPIO, or via serial command.

**CONTACT**  
**Website:** [www.swarm.space](http://www.swarm.space)  
**Email:** [info@swarm.space](mailto:info@swarm.space)

<b>COMPONENTS</b>	GPS, VHF radio with integrated T/R switch, U.FL connector for GPS and VHF antennas, ARM Cortex-M4 processor, indicator LEDs, 3.3 V serial UART interface, 3.3 V GPIO
<b>SENSORS</b>	Onboard GPS (lat/lon/alt), CPU Temperature
<b>DIMENSIONS</b>	51.0 mm x 30.0 mm x 5.3 mm
<b>MASS</b>	9.6 g
<b>POWER</b>	<b>Sleep mode (3.3 V):</b> 80 $\mu$ A (max) <b>Receive mode (3.3 V):</b> 26 mA (typ), 40 mA (max) <b>Transmit mode (3.3 V):</b> 850 mA (typ), 1000 mA (max)
<b>ENVIRONMENT</b>	Operational: -40 C to +85 C Storage: -40 C to +85 C
<b>COMMAND INTERFACE</b>	3.3 V Serial UART
<b>BIT RATE</b>	1 kbps
<b>FREQUENCY</b>	137-138 MHz (downlink) 148-150 MHz (uplink)

© 2022 SWARM TECHNOLOGIES [WWW.SWARM.SPACE](http://WWW.SWARM.SPACE)

Figure 35: Swarm M138 Satellite Modem



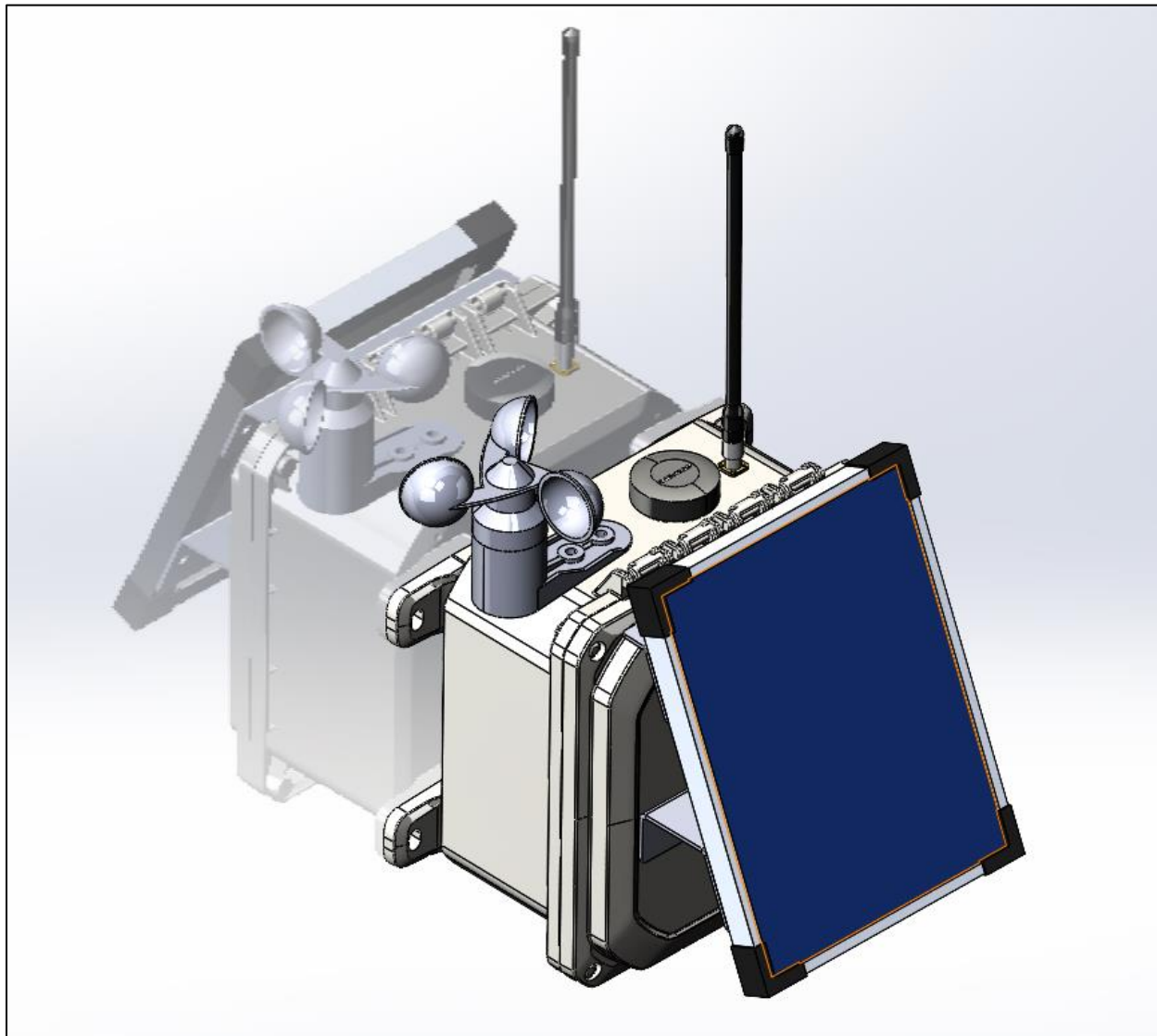


Figure 36: Alternative 3D View

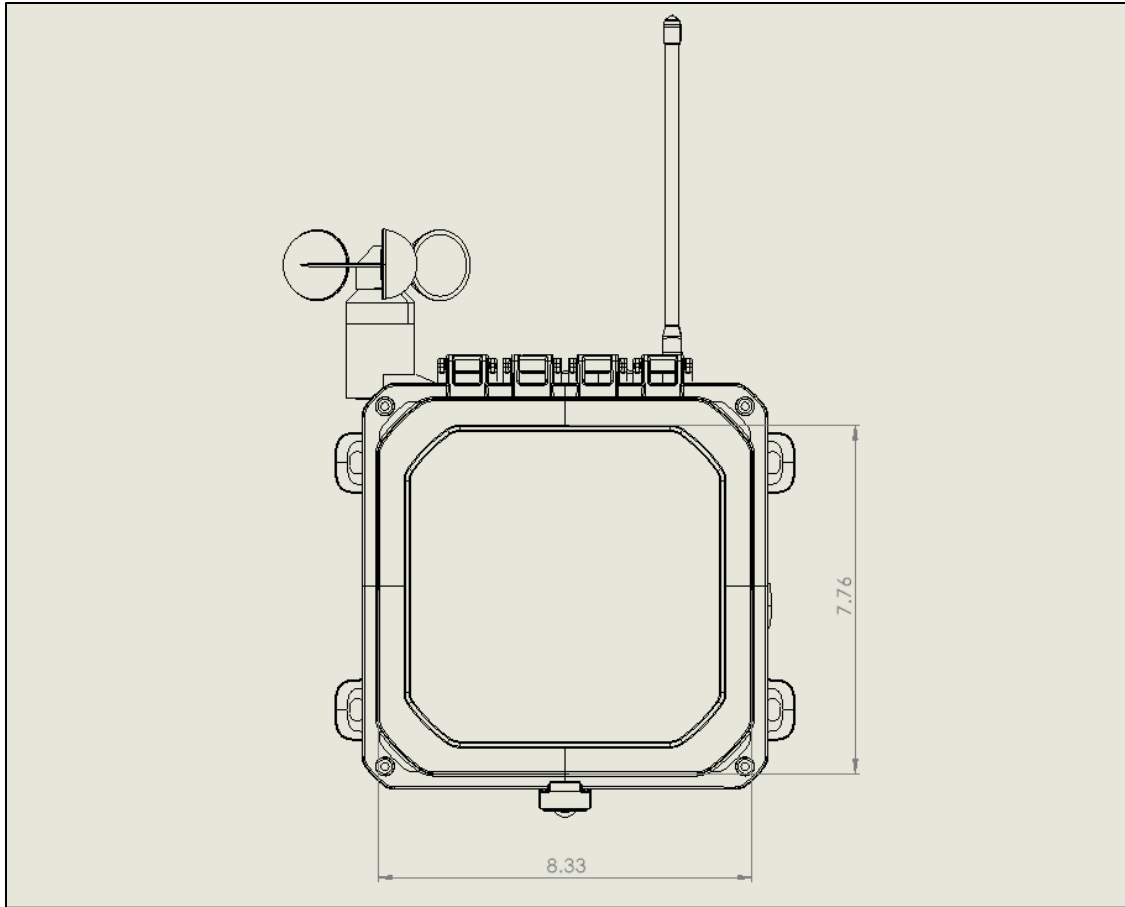


Figure 37: Interior Dimensions<sup>2</sup>

---

<sup>2</sup> All dimension in inches

Table 17: PIC24FJ129GA010 I2C Code Library

```

void I2Cinit(int BRG)
{
    I2C1BRG = BRG;           //See PIC24FJ128GA010 data sheet, Table 16.1
pg. 139
    while (I2C1STATbits.P);  // Check bus idle, see 5.1 of I2C document.

    // It works, not sure its needed.
    I2C1CONbits.A10M = 0;    // 7-bit address mode (Added 8-14-17)
    I2C1CONbits.I2CEN=1;     // enable module
}

void I2CStart(void)
{
    us_delay(10);            // delay to be safe
    I2C1CONbits.SEN = 1;     // Initiate Start condition see pg. 21 of I2C
man. DS70000195F
    while (I2C1CONbits.SEN); // wait for Start condition complete See sec.
5.1
    us_delay(10);            // delay to be safe
}

void I2CStop(void)
{
    us_delay(10);            // delay to be safe
    I2C1CONbits.PEN = 1;     // see 5.5 pg. 27 of Microchip I2C manual
DS70000195F
    while (I2C1CONbits.PEN); // This is at hardware level, & I suspect fast.
    us_delay(10);            // delay to be safe
}

void I2Csendbyte(char data)
{
    while (I2C1STATbits.TBF) ; // wait if buffer is full
    I2C1TRN = data;           // pass data to transmission register
    us_delay(10);            // delay to be safe
}

// Sends master ACK
uint8_t I2Cgetbyte(void)
{
    I2C1CONbits.RCEN =1;     // Set RCEN, Enables I2C Receive mode
    while (!I2C1STATbits.RBF) ; //wait for byte to shift into I2C1RCV register
    I2C1CONbits.ACKEN = 1;    // Master sends Acknowledge
}

```

```

    us_delay(10);                // delay to be safe
    return(I2C1RCV);
}

// Sends master NACK
uint8_t I2Clastgetbyte(void)
{
    I2C1CONbits.RCEN =1;        // Set RCEN, Enables I2C Receive mode
    while (!I2C1STATbits.RBF) ; //wait for byte to shift into I2C1RCV register
    I2C1CONbits.ACKEN = 1;      // Master sends No acknowledge
    I2C1CONbits.ACKDT = 1;
    us_delay(10);                // delay to be safe
    return(I2C1RCV);
}

```

Table 18: NMEA Checksum Code

```

uint8_t nmeaChecksum (const char *sz, size_t len)
{
    size_t i = 0;
    uint8_t cs;
    if (sz [0] == '$')
        i++;
    for (cs = 0; (i < len) && sz [i]; i++)
        cs ^= ((uint8_t) sz [i]);
    return cs;
}

```

Table 19: Raw Pressure to hPa Code

```

double pressure_to_hPa(uint32_t raw_pressure)
{
    // Raw pressure is (uint32_t)pressDataH << 16) | ((uint32_t)pressDataL <<
8) | ((uint32_t)pressDataXL)
    // Where each pressure data packet is 8 bits.
    return (raw_pressure / 4096.0f);
}

```

Table 20: Raw Temperature to °F

```
double temp_to_far(uint16_t raw_temp)
{
    // Raw_temp should be in the format (uint16_t)temp_MSB << 8) |
    ((uint16_t)temp_LSB)
    // Where each temperature packet is 8 bits.
    return (-49 + (315 * (raw_temp / 65535.0f)));
}
```

Table 21: Raw Humidity to Relative Percentage

```
double humid_to_percent(uint16_t raw_humid)
{
    // Raw_temp should be in the format (uint16_t)hum_MSB << 8) |
    ((uint16_t)hum_LSB)
    // Where each humidity packet is 8 bits.
    return (100 * (raw_humid / (65535.0f)));
}
```

Table 22: Demonstration Code

```
#include "mcc_generated_files/system.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Global variables.

int dSec = 0;
int Sec = 0;
int Min = 0;

/* User toggle specifications. Example dSecTimerInterval = 5, secTimerInterval
= 1, minTimerInterval = 1.
*           This will result in the time between BoolToggles
= 1.15 mins.
*/
int dSecTimerInterval = 0; // Desired tenth of a second timer. Set this to 1-
9.
int secTimerInterval = 4; // Desired second timer. Set this to 1-59.
int minTimerInterval = 0; // Desired minute timer. Set this to 1-59.
```

```

// End Global Variables.

// Program Functions
void us_delay(int time)
{
    // Timer 1 On, prescale 1:8, Tcy as clock
    T1CON = 0x8010;

    // Resets TMR1 to 0 to initialize new iteration.
    TMR1 = 0;

    // Delays (does nothing) until TMR1 reaches user inputed delay
    while (TMR1 < (time * 2));
    {
    }
}

// ms_delay function which uses Timer1 to delay a specified amount of time.
void ms_delay(int N)
{
    // Timer 1 On, prescale 1:256, Tcy as clock
    T1CON = 0x8030;

    // Resets TMR1 to 0 to initialize new iteration.
    TMR1 = 0;

    // Delays (does nothing) until TMR1 reaches user inputed delay
    while(TMR1 < N * 62.5);
    {
    }
}

void InitU2(void)
{
    // PIC24FJ128GA010 data sheet, 17.1 for calculation, Fcy = 16MHz. This will
    yield ~115200 baud rate
    U2BRG = 34;

    // See data sheet, pg 148. Enable UART2, BRGH = 1,
    U2MODE =0x8008;

    // Idle state = 1, 8 data, No parity, 1 Stop bit

```

```

    // See data sheet, pg. 150, Transmit Enable
    U2STA = 0x0400;
    // Following lines pertain Hardware handshaking
    // enable RTS , output
    TRISFbits.TRISF13 = 1;
}

char putU2(char c)
{
    // Wait if transmit buffer full.
    while (U2STAbits.UTXBF );

    // Write value to transmit FIFO
    U2TXREG = c;
    return c;
}

char getU2 (void)
{
    // Read from receiving buffer
    return U2RXREG;
}

// NMEA Checksum Function (Swarm Provided)
uint8_t nmeaChecksum (const char *sz, size_t len)
{
    size_t i = 0;
    uint8_t cs;
    if (sz [0] == '$')
        i++;
    for (cs = 0; (i < len) && sz [i]; i++)
        cs ^= ((uint8_t) sz [i]);
    return cs;
}

// Sets GPIO pin to be high when there are unsent messages
// Continue to add more setup commands here as needed.
void m138_setup()
{
    char message_combined[999];
    char swarm_message[999];
    size_t combined_message_length = sprintf(message_combined, "GP 10");

```

```

    // Calculate NMEA checksum
    uint8_t cs = nmeaChecksum(message_combined, combined_message_length);

    size_t message_size = sprintf(swarm_message, "$s*%02x\n",
message_combined, cs);

    // Send message char by char over UART
    int i;
    for(i = 0; i < message_size; i++)
    {
        putU2(swarm_message[i]);
        us_delay(15);
    }
}

// Preconditions: Application ID, Hold Duration of Message, Char of Message
// Function: Formats, calculates checksum, and transmits UART message in
correct form.
void m138_transmit_message(int app_id, int hold_dur, const char *text)
{
    char message_combined[999];
    char swarm_message[999];
    size_t combined_message_length = sprintf(message_combined, "TD
AI=%d,HD=%d,\"%s\"", app_id, hold_dur, text);

    // Calculate NMEA checksum
    uint8_t cs = nmeaChecksum(message_combined, combined_message_length);

    size_t message_size = sprintf(swarm_message, "$s*%02x\n",
message_combined, cs);

    // Send message char by char over UART
    int i;
    for(i = 0; i < message_size; i++)
    {
        putU2(swarm_message[i]);
        us_delay(15);
    }
}

// I2C Related Functions

```



```

void I2Cinit(int BRG)
{
    I2C1BRG = BRG;                //See PIC24FJ128GA010 data sheet, Table 16.1
pg. 139
    while (I2C1STATbits.P);       // Check bus idle, see 5.1 of I2C document.

    // It works, not sure its needed.
    I2C1CONbits.A10M = 0;         // 7-bit address mode (Added 8-14-17)
    I2C1CONbits.I2CEN=1;          // enable module
}

void I2CStart(void)
{
    us_delay(10);                 // delay to be safe
    I2C1CONbits.SEN = 1;          // Initiate Start condition see pg. 21 of I2C
man. DS70000195F
    while (I2C1CONbits.SEN);      // wait for Start condition complete See sec.
5.1
    us_delay(10);                 // delay to be safe
}

void I2CStop(void)
{
    us_delay(10);                 // delay to be safe
    I2C1CONbits.PEN = 1;          // see 5.5 pg. 27 of Microchip I2C manual
DS70000195F
    while (I2C1CONbits.PEN);      // This is at hardware level, & I suspect fast.
    us_delay(10);                 // delay to be safe
}

void I2Csendbyte(char data)
{
    while (I2C1STATbits.TBF) ;    // wait if buffer is full
    I2C1TRN = data;               // pass data to transmission register
    us_delay(10);                 // delay to be safe
}

// Sends ACK from master
uint8_t I2Cgetbyte(void){
    I2C1CONbits.RCEN =1;          // Set RCEN, Enables I2C Receive mode
    while (!I2C1STATbits.RBF) ;  //wait for byte to shift into I2C1RCV register
    I2C1CONbits.ACKEN = 1;        // Master sends Acknowledge
    us_delay(10);                 // delay to be safe
    return(I2C1RCV);
}

```

```

}

// Sends no ACK from master.
uint8_t I2Clastgetbyte(void){
    I2C1CONbits.RCEN =1;           // Set RCEN, Enables I2C Receive mode
    while (!I2C1STATbits.RBF) ; //wait for byte to shift into I2C1RCV register
    I2C1CONbits.ACKEN = 1;         // Master sends No acknowledge
    I2C1CONbits.ACKDT = 1;
    us_delay(10);                  // delay to be safe
    return(I2C1RCV);
}

// PMP initialization. See my notes in Sec 13 PMP of Fam. Ref. Manual
void InitPMP(void)
{
    PMCON = 0x8303;                // Following Fig. 13-34. Text says 0x83BF (it works) *
    PMMODE = 0x03FF;              // Master Mode 1. 8-bit data, long waits.
    PMAEN = 0x0001;               // PMA0 enabled
}

// InitLCD Fucntion to allow interfacing with LCD
void InitLCD(void)
{
    // PMP is in Master Mode 1, simply by writing to PMDIN1 the PMP takes care
    // of the 3 control signals so as to write to the LCD.
    PMADDR = 0;                   // PMA0 physically connected to RS, 0 select
    Control register
    PMDIN1 = 0b00111000;          // 8-bit, 2 lines, 5X7. See Table 9.1 of text
    Function set
    ms_delay(1);                  // 1ms > 40us
    PMDIN1 = 0b00001100;          // ON, cursor off, blink off
    ms_delay(1);                  // 1ms > 40us
    PMDIN1 = 0b00000001;          // clear display
    ms_delay(2);                  // 2ms > 1.64ms
    PMDIN1 = 0b00000110;          // increment cursor, no shift
    ms_delay(2);                  // 2ms > 1.64ms
} // InitLCD

// Function to Read HD44780 Status Register & Macros to supply Status, Control
// & Data Info.
char ReadLCD(int addr)
{
    // As for dummy read, see 13.4.2, the first read has previous value in
    PMDIN1

```

```

    int dummy;
    while (PMMODEbits.BUSY);           // wait for PMP to be available
    PMADDR = addr;                     // select the command address
    dummy = PMDIN1;                    // initiate a read cycle, dummy
    while (PMMODEbits.BUSY);           // wait for PMP to be available
    return (PMDIN1);                   // read the status register
} // ReadLCD
// In the following, addr = 0 -> access Control, addr = 1 -> access Data
#define BusyLCD() ReadLCD( 0) & 0x80   // D<7> = Busy Flag
#define AddrLCD() ReadLCD( 0) & 0x7F   // Not actually used here
#define getLCD() ReadLCD( 1)           // Not actually used here.

// Function to Write to Control Register & Data (i.e., ASCII Characters) &
// Macros
// ASCII characters to Data as well as Control reg. commands.
void WriteLCD(int addr, char c)
{
    while (BusyLCD());
    while (PMMODEbits.BUSY);           // wait for PMP to be available
    PMADDR = addr;
    PMDIN1 = c;
} // WriteLCD
// In the following, addr = 0 -> access Control, addr = 1 -> access Data
#define putLCD( d) WriteLCD( 1, (d))
#define CmdLCD( c) WriteLCD( 0, (c))
#define HomeLCD() WriteLCD( 0, 2)      // See HD44780 instruction set in
#define ClrLCD() WriteLCD( 0, 1)       // Table 9.1 of text book

void putsLCD(char *s)
{
    while (*s) putLCD(*s++);           // See paragraph starting at bottom,
pg. 87 text
} //putsLCD

void SetCursorAtLine(int i)
{
    TRISA = 0;                         // Sets PORTA to output.
    if (i == 1)
    {
        CmdLCD(0x80);
    }else if (i == 2)
    {
        CmdLCD(0xC0);
    }else
    {

```

```

    // Sets flag for while loop and resets global variables to set delay
    timer to 0.
    int flag = 1;
    dSec = 0;
    Sec = 0;
    Min = 0;

    while(flag)
    {
        // Until the global clock reaches the desired time, flash LEDs at
        5Hz, or 100ms Ton and 100ms Toff = 200ms = T.
        // Then when time is reached specified by user, turn off LEDs.
        if ((dSec == dSecTimerInterval) && (Sec == secTimerInterval) &&
        (Min == minTimerInterval))
        {
            PORTA = 0x00;
            flag = 0;        // Clear flag so while loop exits.
        }else
        {
            PORTA = 0xff;    // Set LEDs High.
            ms_delay(100);   // Delay.
            PORTA = 0x00;    // Set LEDs low.
            ms_delay(100);   // Delay.
        }
    }
}

void turnOffDisplay(void)
{
    PMADDR = 0;                // PMA0 physically connected to RS, 0 select
    Control register
    PMDIN1 = 0b00001000;      // 8-bit, 2 lines, 5X7. See Table 9.1 of text
    Function set
    ms_delay(1);
}

void turnOnDisplay(void)
{
    PMADDR = 0;                // PMA0 physically connected to RS, 0 select
    Control register
    PMDIN1 = 0b00001100;      // 8-bit, 2 lines, 5X7. See Table 9.1 of text
    Function set
    ms_delay(1);
}

```

```

}

// Main Function
int main(void)
{
    // Initialize the device and initialize UART baud rate.
    SYSTEM_Initialize();
    InitU2();
    I2Cinit(157);

    InitPMP(); // Initialize the Parallel Master
Port
    InitLCD(); // Initialize the LCD
    char LCDchar[8]; // Create an array for 8 characters

    char swarm_message[999];

    uint8_t pressDataXL = 0;
    uint8_t pressDataL = 0;
    uint8_t pressDataH = 0;
    double pressDataAnz = 0;

    // Set PORTA to output.
    TRISA = 0;

    // Variables for serial communication to Putty.
    int x = 1;

    // While loop for UART debugging that runs once.
    while(1)
    {
        if (x == 1)
        {
            // Insert Code Here (Will only be run once per reset)
            us_delay(100);
            I2Cstart(); // I2C Start condition
            us_delay(100);
            I2Csendbyte(0xBA); // Send Address of 7 seg for write mode
            us_delay(100);
            I2Csendbyte(0x11);
            us_delay(100);
            I2Csendbyte(0b00010000);
        }
    }
}

```

```

        us_delay(100);
        I2CStop();

        us_delay(100);
        I2CStart();                // I2C Start condition
        us_delay(100);
        I2Csendbyte(0xBA);
        us_delay(100);
        I2Csendbyte(0x28);
        us_delay(100);
        I2CStop();

        us_delay(100);
        I2CStart();                // I2C Start condition
        us_delay(100);
        I2Csendbyte(0xBB);
        us_delay(100);
        pressDataXL = I2Cgetbyte();
        us_delay(100);
        pressDataXL = I2Cgetbyte();
        us_delay(100);
        pressDataH = I2Clastgetbyte();
        us_delay(100);
        I2CStop();
        us_delay(100);

        uint32_t pressLSBw = (((uint32_t)pressDataH << 16) |
        ((uint32_t)pressDataL << 8) | ((uint32_t)pressDataXL));
        pressDataAnz = pressLSBw / 4096.0f;

        sprintf(LCDchar, "%0.2f hg", pressDataAnz);
        SetCursorAtLine(1);
        putsLCD("Pressure is : ");                // Put message on first
line of LCD
        SetCursorAtLine(2);                // Set to start of second
line
        putsLCD(LCDchar);                // Floating Point
Temperature on second line.

        sprintf(swarm_message, "Pressure from ASEC 509 is %0.2f hg",
pressDataAnz);

        us_delay(100);
        m138_setup();
        m138_transmit_message(40, 7200, swarm_message);

```

```
        // Delay to allow easy tracking of data.  
        ms_delay(250);  
  
        x = 0;  
    }  
}  
return 0;  
}
```